MacTech Magazine
Vol. 18, No. 08 • August 2002

MacTech®

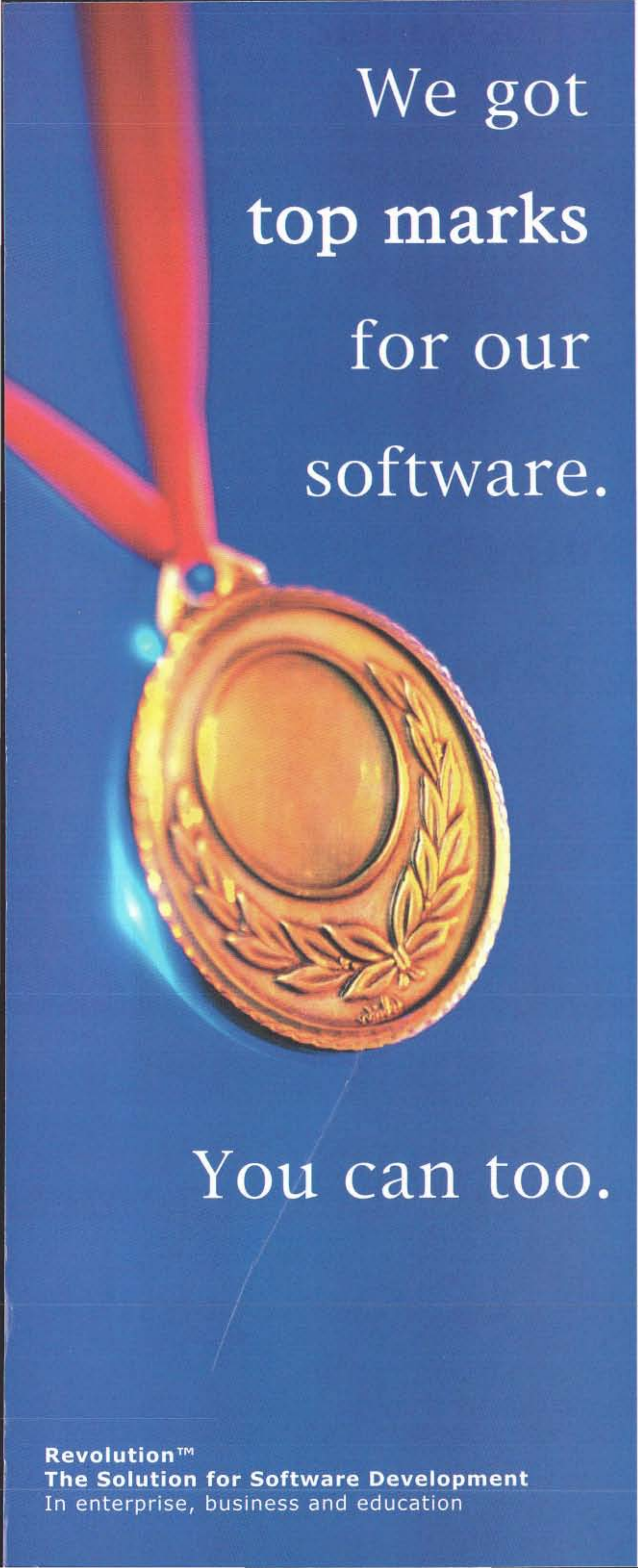
The Journal of Macintosh Technology and Development

Reusable WebObjects Components

By Tom Woteki

Develop reusable, cooperating components using key-value coding

$8.95 US
$12.95 Canada
ISSN 1067-8360
Printed in U.S.A.



REVOLUTION™

Limitless possibilities

With Revolution, the solutions you can create are endless.

True cross-platform development

With the click of a button, you can build applications for Macintosh (Classic and OS X), Windows, Linux, and Unix.

Increased productivity

The powerful interface builder and easy-to-use programming language speed up all the essentials of software development, leaving you with more time to focus on your project.

Databases, multimedia, Internet applications, external libraries and more

Revolution supports everything you need: SQL databases, streaming media, control of QuickTime, QTVR and graphics, CGI processing, Internet protocols, and more.

FREE Trial Version

www.runrev.com

MacUser Awards 2001 Nominated

MacUser UK - 5 mice

Revolution™
The Solution for Software Development
In enterprise, business and education

RUNTIME REVOLUTION

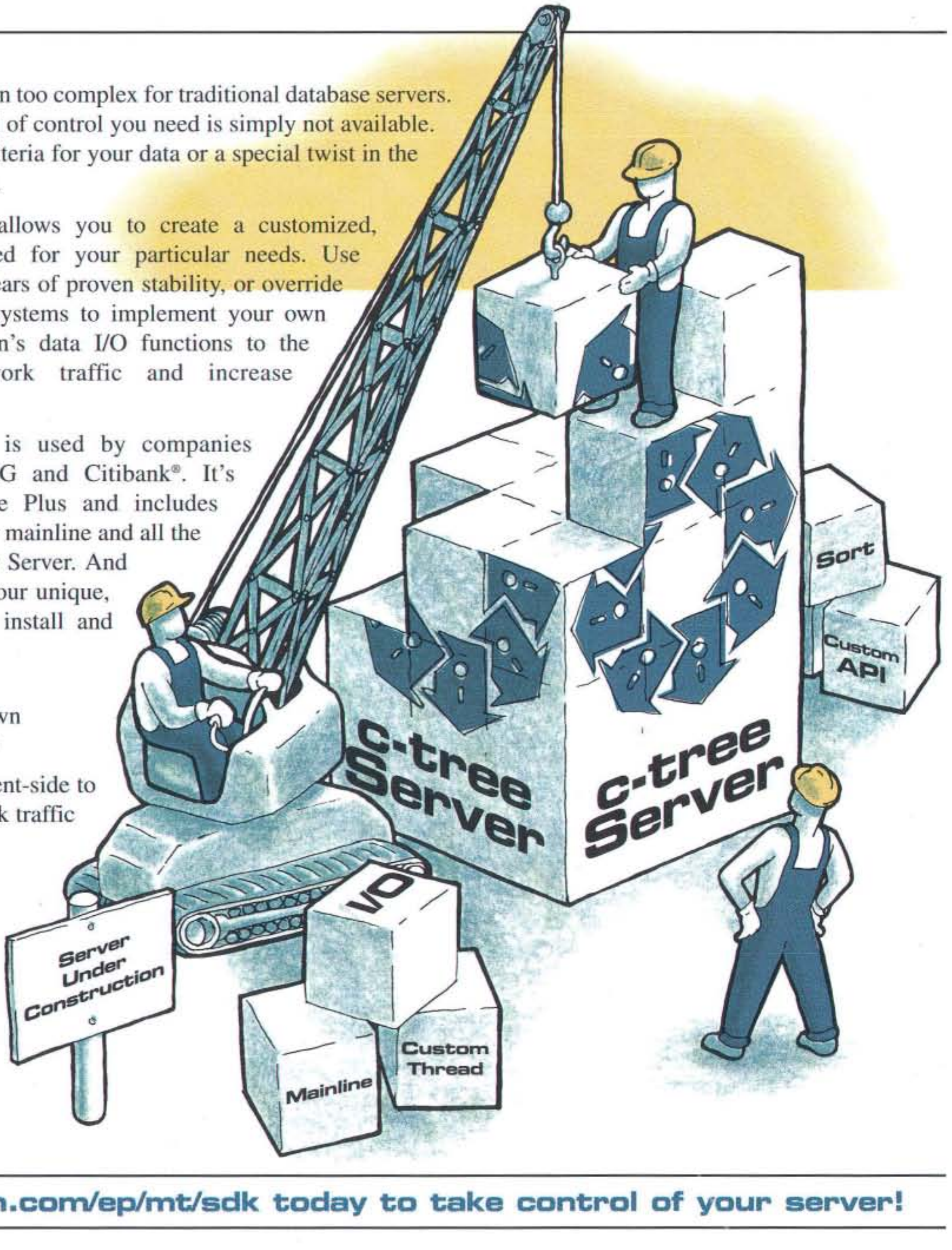
CUSTOMIZE
YOUR DATABASE SERVER WITH THE
C-TREE® SERVER SDK

Today's database demands are often too complex for traditional database servers. The functionality and precise level of control you need is simply not available. Perhaps you need alternate sort criteria for your data or a special twist in the threading or communication logic.

FairCom's c-tree® Server SDK allows you to create a customized, industrial-strength server designed for your particular needs. Use FairCom's kernel, with over 20 years of proven stability, or override functionality within specific subsystems to implement your own subtleties. Move your application's data I/O functions to the server-side to decrease network traffic and increase performance!

FairCom's c-tree Server SDK is used by companies worldwide such as Software AG and Citibank®. It's integrated seamlessly into c-tree Plus and includes complete source code to the server mainline and all the interface subsystems to the c-tree Server. And best of all, once you've created your unique, customized server, it is easy to install and administer: no DBA required!

- Enhance our server with your own custom server-side functionality
- Move functionality from the client-side to the server-side to reduce network traffic and increase performance
- Modify or replace entire server subsystems
- Complete source for the server mainline, key server subsystems, and client-side
- Flexible OEM licensing



Visit www.faircom.com/ep/mt/sdk today to take control of your server!

FairCom®
www.faircom.com

USA	573.445.6833
EUROPE	+39.035.773.464
JAPAN	+81.59.229.7504
BRAZIL	+55.11.3872.9802

DBMS Since 1979 • 800.234.8180 • info@faircom.com

XSERVE SCREAMS.

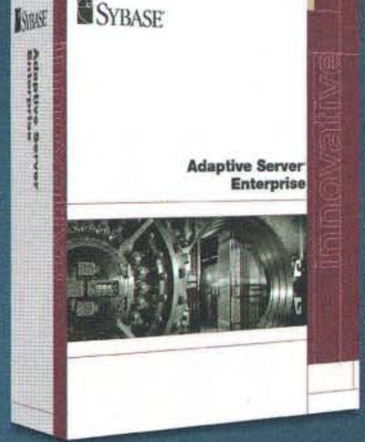
(FOR AN ENTERPRISE CLASS DATABASE.)

An insanely great enterprise server deserves an insanely great enterprise database. Sybase Adaptive Server™ Enterprise 12.5. Wall Street's preferred platform for mission-critical, transaction-intensive

SYBASE e-BUSINESS SOFTWARE. EVERYTHIN

THE STRAIGHT GOODS ON DATABASES.

WE ANSWER THE CALL.



enterprise applications. Coming soon to you on Apple Xserve™ with Mac OS X Server software. Ready to scream for joy? Check it out at **www.sybase.com/mac**

SYBASE®
Information Anywhere™

WORKS BETTER WHEN EVERYTHING WORKS TOGETHER.℠

MacTech®

The Journal of Macintosh Technology & Development

A publication of **XPLAIN** CORPORATION

How To Communicate With Us

In this electronic age, the art of communication has become both easier and more complicated. Is it any surprise that we prefer **e-mail?**

If you have any questions, feel free to call us at 805/494-9797 or fax us at 805/494-9798.

If you would like a subscription or need customer service, feel free to contact Developer Depot Customer Service at 877-MACTECH

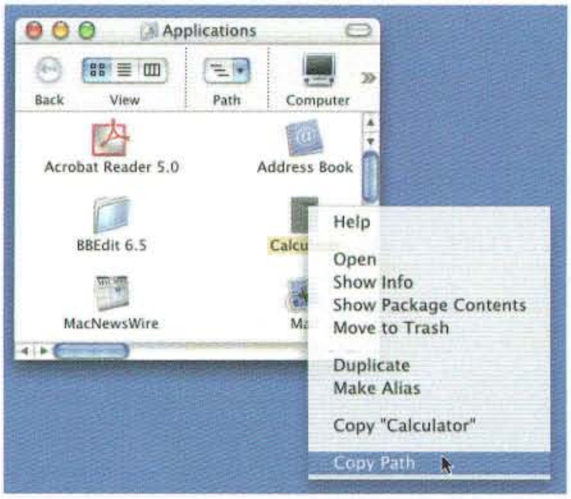
DEPARTMENTS	E-Mail/URL
Orders, Circulation, & Customer Service	cust_service@mactech.com
Press Releases	press_releases@mactech.com
Ad Sales	ad_sales@mactech.com
Editorial	editorial@mactech.com
Programmer's Challenge	prog_challenge@mactech.com
Online Support	online@mactech.com
Accounting	accounting@mactech.com
Marketing	marketing@mactech.com
General	info@mactech.com
Web Site (articles, info, URLs and more...)	http://www.mactech.com

MacTech Magazine (ISSN: 1067-8360 / USPS: 010-227) is published monthly by Xplain Corporation, 850-P Hampshire Road, Westlake Village, CA 91361-2800. Voice: 805/494-9797, FAX: 805/494-9798. Domestic subscription rates are $47.00 per year. Canadian subscriptions are $59.00 per year. All other international subscriptions are $97.00 per year. Domestic source code disk subscriptions are $77 per year. All international disk subscriptions are $97.00 a year. Please remit in U.S. funds only. Periodical postage is paid at Thousand Oaks, CA and at additional mailing office.

POSTMASTER: Send address changes to **MacTech Magazine**, P.O. Box 5200, Westlake Village, CA 91359-5200.

Contents

August 2002 • Volume 18, Issue 08



Copy Path command in Finder's contextual menu . . .page 8

By Rich Morin

A Brief Look at Perl

a real gem of a scripting language

Last month's column closed with some short descriptions of BSD's scripting languages. It offered some opinions, but stopped short of recommending any particular language. This month, I'll get a bit braver, explaining why you might want to use Perl for most of your BSDish scripting needs.

First, however, I should caution that Perl is **not** always the appropriate choice. If you're modifying a system shell script, don't try to rewrite it in Perl. Just use the language the script is written in (typically the Bourne Shell). If you are just mechanizing a simple list of commands, Perl is probably overkill, but see below. Finally, if the script has to run early in the startup process, the Perl interpreter may not be available.

For new, substantial scripts, however, I would strongly recommend that you use Perl. Here are some reasons:

- efficiency – The Perl interpreter, unlike the shells, seldom has to start up new processes. This means that substantial Perl scripts will often run much faster than equivalent shell scripts. I found this out several years ago, when I transliterated several large shell scripts to Perl. The run times went down by a factor of five!
- syntax – Most shells have very weak notions of syntax. One result of this, in Mac OS X, is that white space in file names can be interpreted as splitting the names into multiple tokens. Perl handles strings in a much more sophisticated manner, so it doesn't get confused.
- integration – Unlike shell scripts, which may stitch together dozens of commands, Perl is an integrated language. This eliminates a great deal of hassle and possible confusion.
- facilities – Perl has powerful data structures, convenient control-flow operators, and access to almost any imaginable system call. The shells have none of these features. As a result, a Perl program can often do things that would be essentially impossible in any shell.
- support – Perl has numerous books, a vast library of modules, and a very active user community. Most shells have few to none of these resoueces.
- portability – Perl scripts can be run on essentially any modern operating system. With a little forethought, they can run unmodified on several different systems. The shells, in contrast, only work on BSD and other Unix-like systems.

Having said all of this, perhaps I should tell you some of the bad news about Perl:

- complexity – Perl is a very large language, with some really peculiar nooks and crannies. Even if you don't use all of these features, you may well encounter them in a module or some other bit of code you "inherit".
- informality – Perl's motto ("There's More Than One Way To Do It" gives fair warning that this isn't a nice tidy "orthogonal" language. In fact, Larry Wall (Perl's creator) says that Perl is a "diagonal" language; cutting across the middle often speed things up!
- mutability – Unlike the shells, Perl is still evolving. Perl 5 has (mostly) stabilized, but Perl 6 development is quite active. So, you might need to relearn some things in a few years.

SHOW ME SOME CODE!

This being MacTech, you're probably wondering when you're going to see some actual Perl code. Well, here's a short Perl script that I hacked together to do some backups. It's not a full-featured backup utility, by any means, but it gets the job done (and shows off some Perl language features)...

```
#!/usr/bin/env perl
#
# macbac - Create backup files, using tar(1).
#
# Written by Rich Morin, CFCL, 2002.06

{
  $date = `date +%y%m%d.%H%M`;
  chomp($date);

  @dir = ('/Users/rdm',
          '/Volumes/Work');

  for $dir (@dir) {
    $bac = cvt($dir);
    $cmd = "nice -10 tar czf $bac $dir";
    printf(">>> %s\n", $cmd);
    system($cmd);
  }
}

sub cvt {  # convert the directory name
  my ($tmp) = @_;

  $tmp =~ s|/|.|g;
  $tmp =~ s|\s|_|g;
  return("/Backups/$date$tmp.tgz");
}
```

The first line of any BSD script, as discussed previously, tells the system which program should be invoked as the interpreter

Rich Morin has been using computers since 1970, Unix since 1983, and Mac-based Unix since 1986 (when he helped Apple create A/UX 1.0). When he isn't writing this column, Rich runs Prime Time Freeware (www.ptf.com), a publisher of books and CD-ROMs for the Free and Open Source software community. Feel free to write to Rich at rdm@ptf.com.

for the following lines. Because I may decide to install a later version of the Perl interpreter at some point (e.g., in /usr/local/bin), I don't want to specify a full path name for the Perl interpreter. So, I tell the system to run /usr/bin/env, letting it find and run the appropriate version of Perl.

The remainder of the script, in any case, is read by the Perl interpreter. Perl's syntax and feature set are borrowed from a variety of (mostly Unix) languages and tools, including awk, Basic-Plus, C, sed, sh, and tr. This makes Perl seem familiar to Unix aficionados, but can cause some culture shock to others. Stay calm; it's not really all that bad!

Unlike C, Perl has no "block comments". So, I use a column of sharp signs (#) for my header comments. I also like to wrap the "main" routine in braces. This causes its contents to be indented at the same level as the contents of any sub (routine). It also gives me a visual cue that this is a "block" of code.

I could have asked Perl to grab and format the date information (and should have, if I were trying for cross-OS portability), but the method above shows off Perl's ability to run BSD commands and retrieve their results. The backquotes tell Perl to run the enclosed command, returning the result as a text string. The result gets put into a scalar variable, $date. The chomp() function, by the way, removes the trailing newline from date's output.

The script then tells Perl to create an array variable named @dir and load it with a list containing two text strings. I use single quotes to wrap these strings, indicating that I don't want Perl to do any variable interpolation (see below) or other tricks.

The for loop sets $dir, successively, to each of the values in @dir. Note that the sigil (e.g., $, @) is part of a Perl variable's name, so @dir and $dir are two different variables. This seems a bit weird at first, but ends up being quite handy as you get used to it.

The cvt() routine shows off some of Perl's capabilities and peculiarities. First, it grabs @_ (the array of calling parameters) and copies the contents into a private list of variables. In this case, the list only has one element, but it might well have more.

The next two lines tell Perl to do global substitutions of periods for slashes and underscores for "white space". This gives me "flattened" names (no directory levels) without any annoying spaces, tabs, etc. This is a trivial example of Perl's powerful "regular expression" capability. Regular expressions can be used to perform all sorts of magic on text strings. In fact, there is a substantial book on regular expressions alone!

The last line tells Perl to "interpolate" the variables $bac and $date into a text string. The use of double quotes tells Perl to look for dollar signs and other "magic" characters. Note that, although $tmp is a private variable, $date is shared with the main routine. Finally, the return is not strictly needed (the value of the last expression evaluated in a sub is automatically returned), but I think it adds to the clarity of the code.

Returning to the main routine, we build up a command string, print it (ala C), and hand it off to the operating system to be run. Look up the man pages for nice and tar to see what their roles are in this script.

In a script of this size, there isn't much room to get into Perl's fancier aspects. Next month, I'll give you a more substantial taste of its data structures and control flow, as well as listing some useful Perl resources. If you can't wait to get started, however, just bop over to www.perl.{org,com}...

New PRIMEBASE 4.0 Connectivity

PHP REALbasic

PRIMEBASE SQL DATABASE SERVER CONNECTIVITY

- REALbasic Plugin
- PHP 4
- JDBC
- ODBC
- EOF
- Perl
- WebObjects
- PrimeBase Application Server
- Witango
- Lasso
- 4D Plugin
- Omnis Studio
- C-API libraries

[get a developer key free of charge]

download now!
www.primebase.com

AVAILABLE ON THE MOST POPULAR PLATFORMS

- Completely cross-platform
- Full-text searching and indexing
- Mac OS classic & X
- Mac OS X Server
- Linux
- Solaris
- IBM AIX
- all Windows platforms

Develop on a Mac and deploy without any additional effort on any platform you want.

PRIMEBASE

SNAP Innovation GmbH
Altonaer Poststraße 9a
D-22767 Hamburg / Germany
www.primebase.com
e-mail: info@primebase.com
Fon: ++49 (40) 389 044-0
Fax: ++49 (40) 389 044-44

By Brent Simmons

Writing Contextual Menu Plugins for OS X, part 1

Implementing the new COM-compatible API

The API for contextual menu plugins is different in OS X than in OS 8/9. It's COM-compatible—it's based on Microsoft's Component Object Model.

That should be enough to scare you off, but don't let it. The good news is you can deal with the COM interface once and re-use that code. The even better news is that this article will provide it for you so you won't have to write it in the first place.

After getting the COM parts out of the way, this article will show how to add a Copy Path menu command to the Finder's contextual menu. This command will copy the path of a selected file or folder onto the clipboard (a path as in /Users/john/myCat.tiff). The second article in this series will go further: it will show how to add submenus, work with text selections, and execute Unix commands.

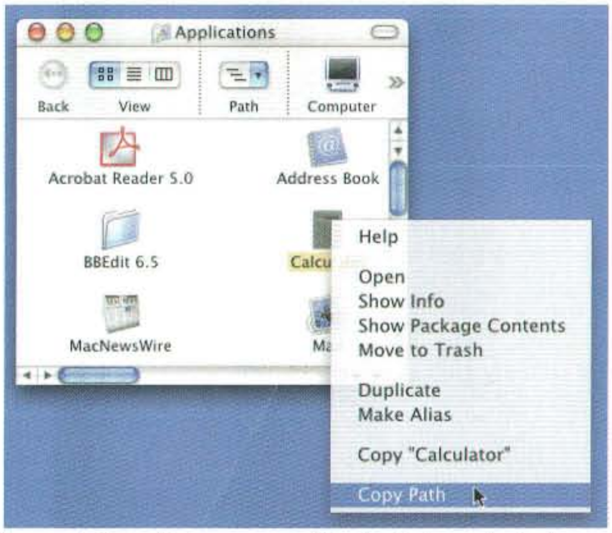


Figure 1. *Copy Path command in Finder's contextual menu*

You can download the source and project files for this article from

http://ranchero.com/downloads/mactech/CopyPathPluginSource.sit

The sample plugin is built with the April 2002 Developer Tools version of Project Builder.

Before we start coding, let's get the mile-high view of contextual menus on OS X.

Overview

The basic gist of contextual menus is that when you control-click on something—some files, a folder, some text—a menu appears with commands that do something with what you clicked on.

In other words, contextual menus act on a selection. You wouldn't, for example, put the Sleep command in a contextual menu, since Sleep doesn't act on a selection.

Also, you wouldn't put an Eject command in a menu that appears when text is selected. Though the Eject command acts on a selection, it doesn't act on text. An Eject command (or similar) should appear only when a removable disk is selected.

Contextual menu plugins are system-wide—almost, anyway. Not every app supports system contextual menus. But you'll find plenty that do, including Eudora, CodeWarrior IDE, BBEdit, Internet Explorer, and of course the Finder.

Plugins are bundles; they're CFPlugins. They can be installed at ~/Library/Contextual Menu Items/ or at /Library/Contextual Menu Items/. The first location makes a plugin accessible to a specific user; the second location makes a plugin accessible to all users.

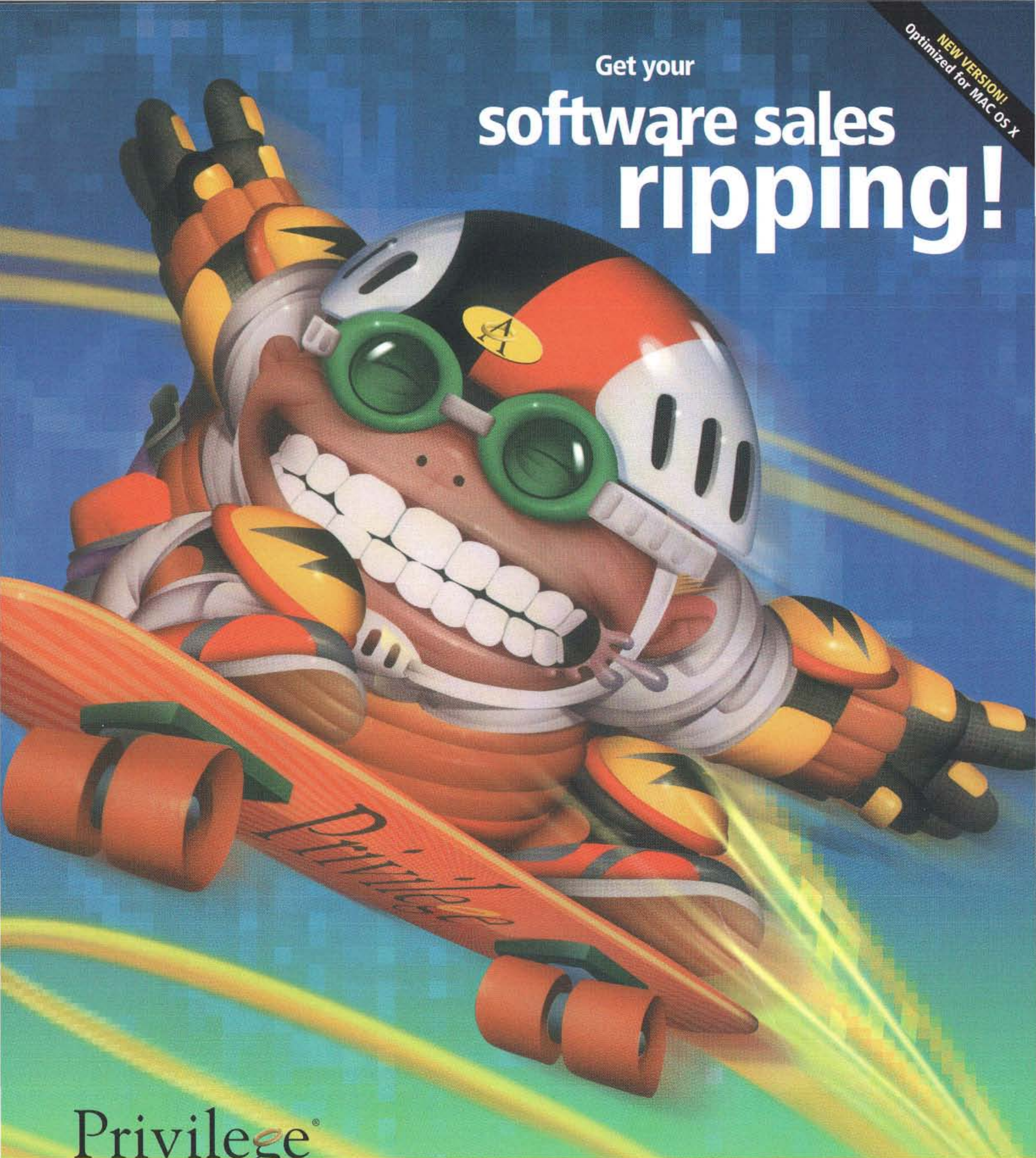
Okay. Let's code.

Implementing the COM IUnknown Interface

A contextual menu plugin has three things it must do: add menu items when requested, run a chosen menu command, and satisfy the requirements of the COM-compatible API.

We'll start with the COM stuff, because it's not utterly thrilling—and, gotten out of the way once means it's out of the way forever. In fact, if you're in a hurry you can skip down to

Brent Simmons is a Seattle-based independent Mac OS X developer and writer, a fan of both Cocoa and Carbon. He runs a Mac developer news weblog at ranchero.com; he can be contacted at brent@ranchero.com.



Privilege®

Tear up that road to market! Allow customers to download your software via the Internet — faster and safer — with the Privilege Software Commerce Solution. Secure trial versions and flexible licensing models that increase your revenue while you simultaneously decrease your operational costs.

To learn why five of the six leading publishers have chosen Aladdin's Privilege to protect, distribute and sell their software over the Internet — zip over to **eAladdin.com/Privilege**. For the full blazin' story, call **1-800-562-2543** and start ripping!

Aladdin®
SECURING THE GLOBAL VILLAGE
eAladdin.com

the UUIDs section below.

IUnknown

Think of COM as a framework for object-oriented plugins. Every plugin must implement the IUnknown interface. IUnknown is the base class; it implements interface querying and reference counting.

Specifically it implements three functions: addRef, which increments the reference count; release, which decrements the reference count and releases the object if the count goes to zero; and queryInterface, which is called to determine if a plugin implements a given interface.

Cocoa developers especially are familiar with reference counting. Each time a COM object is retained its reference count is incremented. When it's released, the reference count is decremented. When it goes to zero the object is disposed.

If you're not familiar with reference counting, you can think of it as a "friends" counter. Each time you get an addRef call, it's like someone ringing you up and saying, hey, I'm your friend. Each time you get a release call, it's like someone ringing you up and saying, hey, I'm no longer your friend. You keep a count of how many friends you have. When the count goes to zero you may as well not exist anymore. (It's a good thing people aren't COM objects.)

Incrementing the reference count

addRef, the first of the required IUnknown interface functions, is called to increment the reference count:

Listing 1: incrementing the reference count

addRef

```
static ULONG addRef (void *pluginInstance)
   {
      tyPlugin *instance = (tyPlugin*) pluginInstance;

      (*instance).refCount += 1;
      return ((*instance).refCount);
   }
```

It gets a pointer to an instance of the plugin and bumps refCount. (Note that though it returns the reference count, this is just a convention, not something to be relied on.)

tyPlugin is a struct defined in CopyPathPlugin.h:

```
typedef struct tyPlugin {
  ContextualMenuInterfaceStruct *cmInterface;
  CFUUIDRef factoryId;
  UInt32 refCount;
  } tyPlugin;
```

The first element of the tyPlugin struct is a pointer to a struct that lays out the functions implemented by the plugin. factoryId is the unique ID of the plugin. refCount is the current reference count.

ContextualMenuInterfaceStruct, also defined in CopyPathPlugin.h, looks like this:

```
static ContextualMenuInterfaceStruct interfaceTable = {
  NULL,
  queryInterface,
  addRef,
  release,
  examineContext,
  handleSelection,
  postMenuCleanup
  };
```

The first item is some necessary padding. The next three functions—queryInterface, addRef, and release—are required by the IUnknown interface. The final three functions—examineContext, handleSelection, and postMenuCleanup—are required by the contextual menus plugin interface.

Decrementing the reference count

release, the second of the required IUnknown interface functions, is called to decrement the reference count. When the reference count goes to zero, the instance is disposed.

Listing 2: decrementing the reference count

release

```
static ULONG release (void *pluginInstance)
   {
      tyPlugin *instance = (tyPlugin*) pluginInstance;

      (*instance).refCount -= 1;

      if ((*instance).refCount == 0) {
        deallocateInstance (instance);
        return (0);
        }

      return ((*instance).refCount);
   }
```

The important part is where it checks if refCount has gone to zero and in that case calls deallocateInstance.

Listing 3: deallocating an instance of the plugin

deallocateInstance

```
static void deallocateInstance (tyPlugin *pluginInstance)
   {
      CFUUIDRef factoryId = (*pluginInstance).factoryId;

      free (pluginInstance);

      if (factoryId) {
        CFPlugInRemoveInstanceForFactory (factoryId);
        CFRelease (factoryId);
        }
   }
```

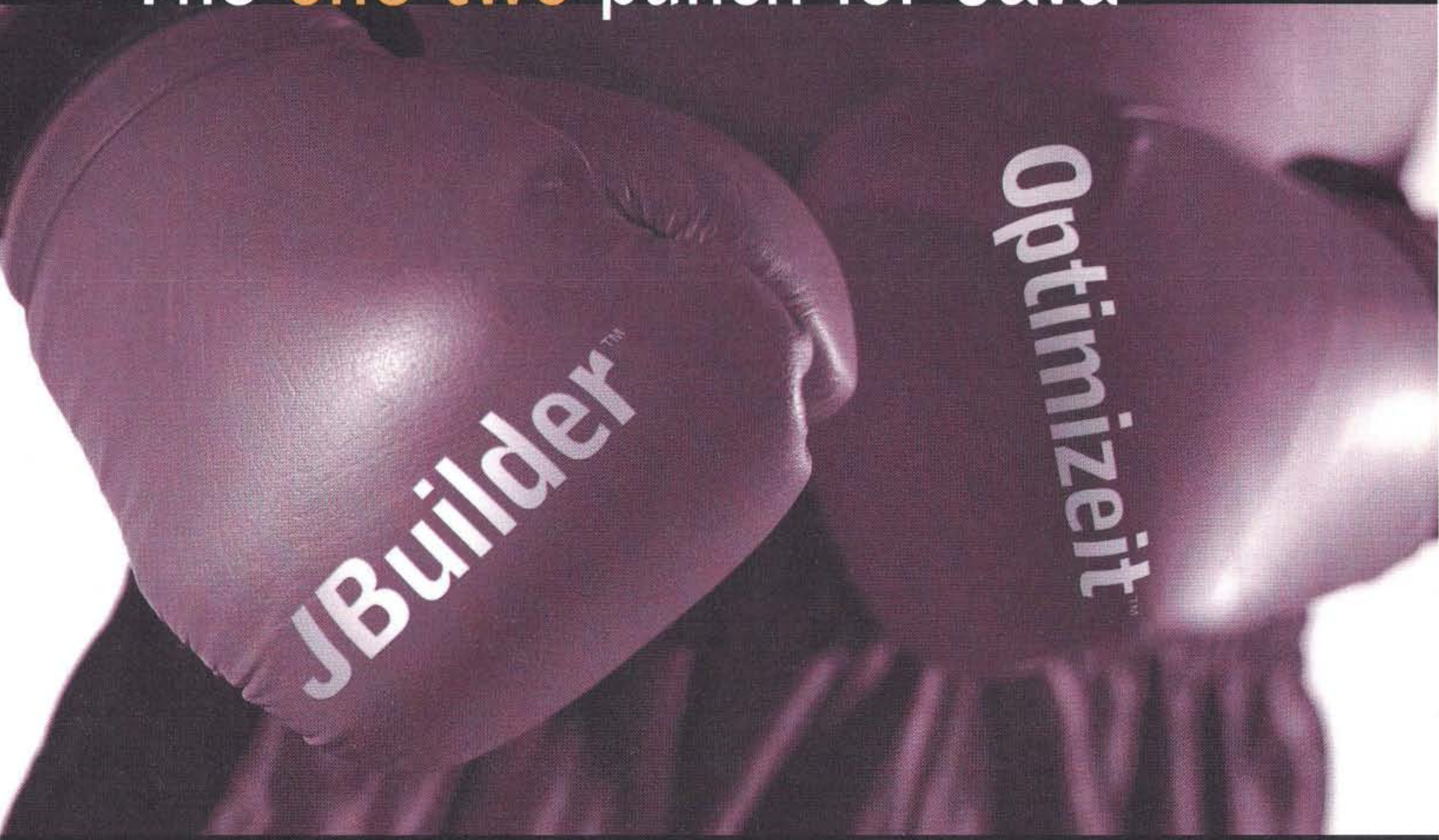
The instance is passed to free which deallocates the memory. Then the instance is disassociated from its factory ID and the factory ID reference is released.

Allocating a new instance

How are instances allocated in the first place? The system calls the factory function pluginFactory to create new instances of the plugin object. It knows to call pluginFactory because in the Application Settings in Project Builder we've specified pluginFactory as the name of the factory method.

Choose Edit Active Target from the Project menu; click on the Bundle Settings tab; click on the Expert button (in the upper right of the window). Expand the CFPluginFactories line and underneath you'll see the UUID of this plugin with a value of pluginFactory.

The one-two punch for Java™



Powering Java development

A winning combination
Successful developers use Borland® JBuilder™ 7 and Borland Optimizeit™ Suite 4.2 together.

Build heavy-hitting Java applications
JBuilder is the leading environment for building Java applications. Rapidly develop Web Services and database applications. The highly productive JBuilder environment includes refactoring, unit testing, team development, and more.

Focus on lightning-fast, reliable code
Optimizeit Suite is an essential for managing performance. Efficiently address speed, memory, and thread issues, so the applications you deploy will scale in production.

Borland®

Find out more about JBuilder 7 and Optimizeit Suite 4.2 at
www.borland.com/new/jb7/5070.html
or call 1-800-252-5547 (offer code: 5070)

Here's pluginFactory itself:

Listing 4: creating a new instance of the plugin

pluginFactory

```
void* pluginFactory (CFAllocatorRef allocator,
  CFUUIDRef typeId)
  {
    #pragma unused (allocator)
    if (CFEqual (typeId, kContextualMenuTypeID))
      return (allocateInstance (kCMPlugInFactoryId));
    return (NULL);
  }
```

First it does a sanity-check (via CFEqual) that what's wanted is a contextual menu plugin, then it calls allocateInstance to actually allocate a new instance.

Listing 5: allocating a new instance of the plugin

allocateInstance

```
static tyPlugin* allocateInstance (CFUUIDRef factoryId)
  {
    tyPlugin *newInstance;

    newInstance = (tyPlugin*) malloc (sizeof (tyPlugin));
    (*newInstance).cmInterface = &interfaceTable;
    (*newInstance).factoryId = CFRetain (factoryId);
    CFPlugInAddInstanceForFactory (factoryId);
    (*newInstance).refCount = 1;

    return (newInstance);
  }
```

Malloc allocates the memory for a new instance, then the elements of the struct are filled in. cmInterface gets a pointer to the interface table (a ContextualMenuInterfaceStruct). The factory ID is set and retained via CFRetain. The instance is associated with this factory ID via CFPlugInAddInstanceForFactory. refCount is set to 1 and the instance is returned.

queryInterface

queryInterface, the third and last of the required IUnknown interface functions, is called to check to see if a given plugin implements a given interface.

Listing 6: handling an interface query

queryInterface

```
static HRESULT queryInterface (void* pluginInstance,
  REFIID iid, LPVOID* ppv)
  {
    if (isGoodInterface (iid)) {
      addRef (pluginInstance);
      *ppv = pluginInstance;
      return (S_OK);
      }

    *ppv = NULL;
    return (E_NOINTERFACE);
  }
```

If the interface is one that this plugin implements, then the reference count is incremented (via addRef) and a pointer to the plugin instance is returned. Otherwise an error code (E_NOINTERFACE) is returned.

isGoodInterface checks to see that the requested interface is either IUnknown or the contextual menu plugin interface.

Listing 7: determining if a requested interface is implemented by this plugin

isGoodInterface

```
static Boolean isGoodInterface (REFIID iid)
  {
    CFUUIDRef interfaceId =
      CFUUIDCreateFromUUIDBytes (NULL, iid);
    Boolean flGoodInterface = false;

    if (CFEqual (interfaceId, kContextualMenuInterfaceID))
      flGoodInterface = true;
    if (CFEqual (interfaceId, IUnknownUUID))
      flGoodInterface = true;

    CFRelease (interfaceId);
    return (flGoodInterface);
  }
```

It creates a CFUUIDRef that can be compared to the contextual menu interface ID and the IUnknown ID. If it matches either interface ID, then it's an interface that this plugin implements. isGoodInterface returns true if there's a match, false otherwise.

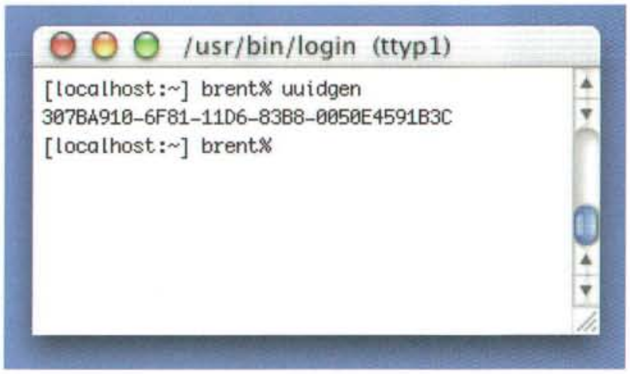
UUIDs

This is the only COM thing you have to re-do for each plugin you create. Luckily, it's a piece of cake, just slightly tedious.

A UUID is a universally unique identifier that identifies your plugin. In this sample it appears in CopyPathPlugin.h as the following un-aesthetic lines:

```
/*The unique ID for this plugin:
CEEFF462-6C32-11D6-A15C-0050E4591B3C*/

#define kCMPlugInFactoryId (CFUUIDGetConstantUUIDWithBytes \
  (NULL, 0xCE, 0xEF, 0xF4, 0x62, 0x6C, 0x32, 0x11, 0xD6, \
  0xA1, 0x5C, 0x00, 0x50, 0xE4, 0x59, 0x1B, 0x3C))
```

To generate a new UUID, launch Terminal.app, and on the command line type uuidgen and hit Return. Underneath will appear a new UUID. (See **Figure 2**.)

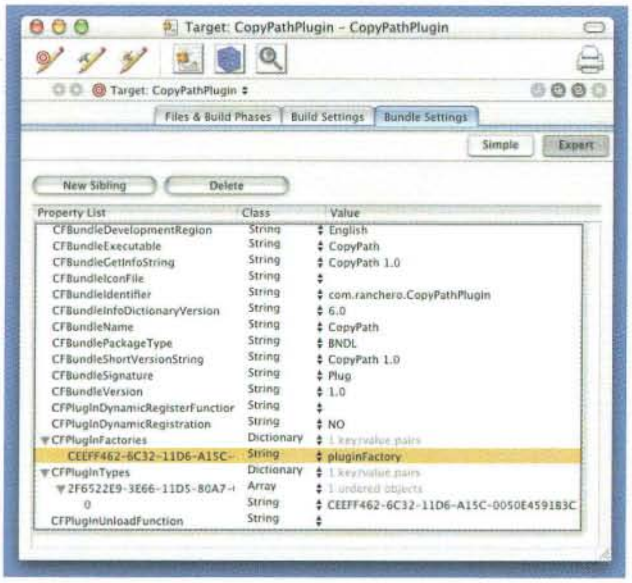


```
[localhost:~] brent% uuidgen
307BA910-6F81-11D6-83B8-0050E4591B3C
[localhost:~] brent%
```

***Figure 2**. Generating a UUID via the command line*

When you go to create your own plugins, copy the UUID generated by uuidgen into your code as in the sample. Then do the tedious bit of copy-and-paste needed to make it look like the above.

Then choose Edit Active Target from the Project menu; click the Bundle Settings tab; click the Expert button. Your UUID should appear in two places (as in this sample): under CFPlugInFactories and under CFPlugInTypes. (See **Figure 3**.)



Property List	Class	Value
CFBundleDevelopmentRegion	String	English
CFBundleExecutable	String	CopyPath
CFBundleGetInfoString	String	CopyPath 1.0
CFBundleIconFile	String	
CFBundleIdentifier	String	com.ranchero.CopyPathPlugin
CFBundleInfoDictionaryVersion	String	6.0
CFBundleName	String	CopyPath
CFBundlePackageType	String	BNDL
CFBundleShortVersionString	String	CopyPath 1.0
CFBundleSignature	String	Plug
CFBundleVersion	String	1.0
CFPlugInDynamicRegisterFunctior	String	
CFPlugInDynamicRegistration	String	NO
▼CFPlugInFactories	Dictionary	1 key/value pairs
CEEFF462-6C32-11D6-A15C-	String	pluginFactory
▼CFPlugInTypes	Dictionary	1 key/value pairs
▼2F6522E9-3E66-11D5-80A7-	Array	1 ordered objects
0	String	CEEFF462-6C32-11D6-A15C-0050E45918 3C
CFPlugInUnloadFunction	String	

***Figure 3**. Bundle settings*

Enough COM. Let's do the fun part, the contextual menus interface.

CONTEXTUAL MENUS INTERFACE

Contextual menu plugins must implement three functions beyond the three required IUnknown interface functions.

examineContext is called to give the chance for the plugin to add commands to the menu that's being built. The plugin can see what the current context is, and decide what commands to add or not add.

handleSelection is called to actually run the command the user chose.

postMenuCleanup is called afterward in case the plugin has any cleanup to do.

examineContext

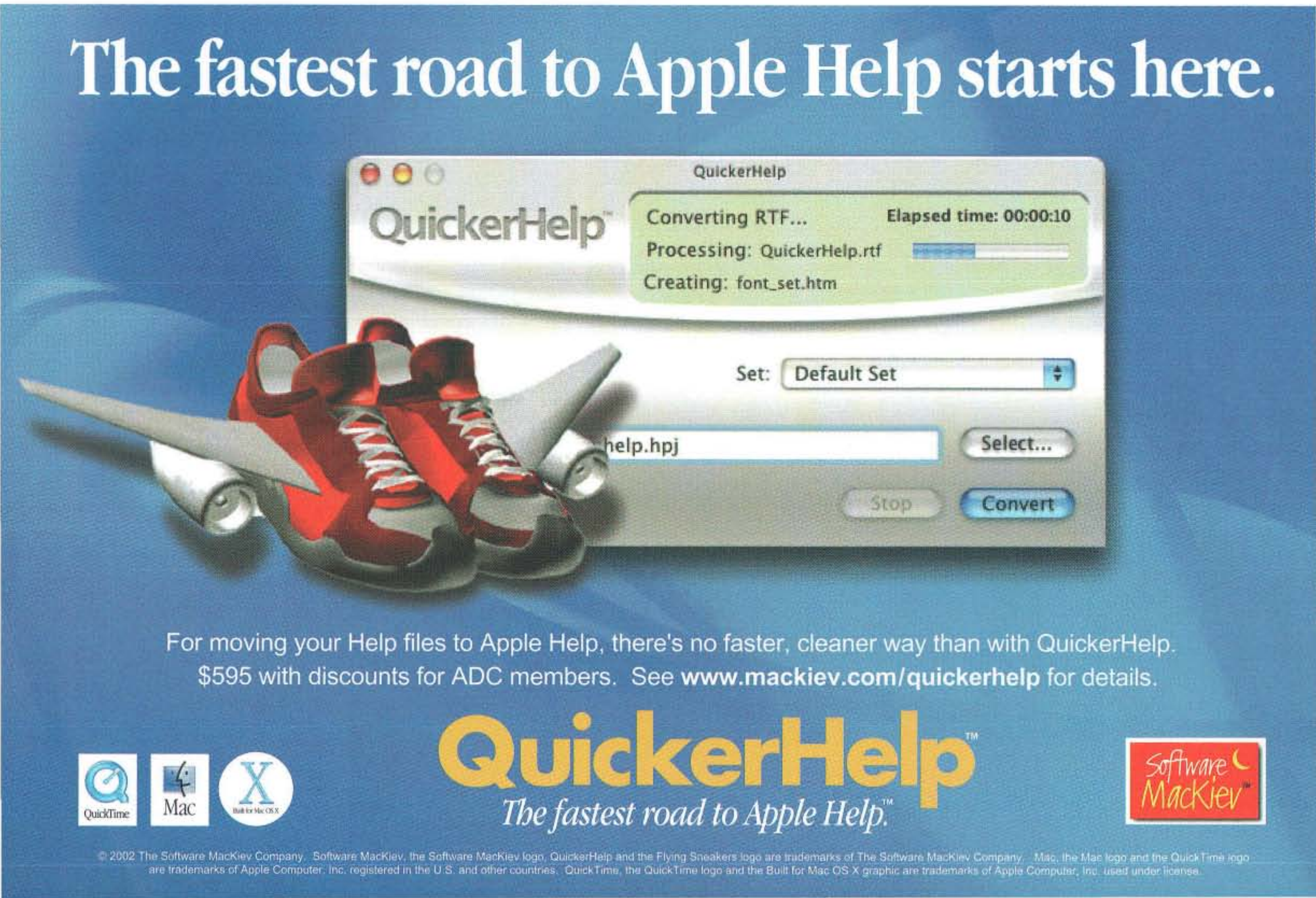
You don't own the contextual menu, you just get the chance to add one or more commands. It's a shared menu: the system may add commands and other plugins may add commands.

This sample plugin will add a Copy Path command when a file or folder is selected in the Finder. examineContext is the first of the required contextual menus interface functions.

Listing 8: determining if a file object is selected

examineContext

```
static OSStatus examineContext (void *pluginInstance,
  const AEDesc *context, AEDescList *commandList)
```



The fastest road to Apple Help starts here.
QuickerHelp
QuickerHelp
Converting RTF...
Elapsed time: 00:00:10
Processing: QuickerHelp.rtf
Creating: font_set.htm
Set: Default Set
help.hpj
Select...
Stop
Convert
For moving your Help files to Apple Help, there's no faster, cleaner way than with QuickerHelp.
$595 with discounts for ADC members. See www.mackiev.com/quickerhelp for details.
QuickerHelp™
The fastest road to Apple Help.™
QuickTime
Mac
Software MacKiev
© 2002 The Software MacKiev Company. Software MacKiev, the Software MacKiev logo, QuickerHelp and the Flying Sneakers logo are trademarks of The Software MacKiev Company. Mac, the Mac logo and the QuickTime logo are trademarks of Apple Computer, Inc. registered in the U.S. and other countries. QuickTime, the QuickTime logo and the Built for Mac OS X graphic are trademarks of Apple Computer, Inc. used under license.

```
  {
    if (getFileObject (context, nil))
      return (addCommandToMenu (commandList));
    if (getFileObjectFromList (context, nil))
      return (addCommandToMenu (commandList));

    return (noErr);
  }
```

Here's an important point: though we said above that we want this command to appear only in the Finder, in fact we'll do something smarter. Let's not care if it it's the Finder or not, let's care only that it's a file or folder object that's selected.

That way, if someone writes a Finder replacement that supports system contextual menus, this plugin will work there too.

A general rule of thumb is that the important part of the context is the type or types of objects selected. The application is of lesser importance, when it's even important at all. In this sample we don't care if it's Apple's Finder, John Doe's cool replacement Finder, or any other app that presents file objects.

context is an Apple event descriptor describing the object or objects selected. What we want to know is if it's a file or folder selected. To do that we see if the object either is or can be coerced to a descriptor of typeAlias (which points to a file or folder).

So examineContext calls getFileObject which returns true if context is already of typeAlias or can be coerced to typeAlias. (It may be of typeFSS, for instance.)

If getFileObject returns false, then examineContext calls getFileObjectFromList. If context is an AEDescList, then it checks to see if the first item in the list is of typeAlias or can be coerced to typeAlias. The reason we do this is because in testing we discovered that context is always an AEDescList when selecting even just one file or folder in the Finder. We could perhaps have gotten away with assuming that that would always be true, and skipped the call to getFileObject, but then a future version of the Finder might break our plugin. (Or the plugin might not work with other apps where context isn't always an AEDescList.)

getFileObject

This function checks to see if a descriptor is of typeAlias or can be coerced to typeAlias.

Listing 9: getting a file object from the context descriptor

getFileObject

```
static Boolean getFileObject (const AEDesc *desc,
  AEDesc *resultDesc)
  {
    AEDesc tempDesc = {typeNull, NULL};
    Boolean flFile = false;
    OSErr err = noErr;

    if ((*desc).descriptorType == typeAlias) {
      if (resultDesc != nil)
        return (AEDuplicateDesc (desc, resultDesc) == noErr);
      else
        return (true);
      }

    err = AECoerceDesc (desc, typeAlias, &tempDesc);
    if ((err == noErr) &&
      (tempDesc.descriptorType == typeAlias)) {
      flFile = true;

      if (resultDesc != nil)
        flFile =(AEDuplicateDesc (&tempDesc, resultDesc)
          == noErr);
      }

    AEDisposeDesc (&tempDesc);
    return (flFile);
  }
```

If context is already of typeAlias, then we know it's a file or folder, and the function returns true. The function checks (*desc).descriptorType to determine the type of the context descriptor.

If context can be coerced to typeAlias (via AECoerceDesc), then again the function returns true. Otherwise it returns false.

Note that this function will make a copy of context (via AEDuplicateDesc) in resultDesc if the caller wants. This will be used later when the plugin is called to run our menu command (via handleSelection).

getFileObjectFromList

This function gets a file object from a list (an AEDescList).

Listing 10: getting a file object from an AEDescList

getFileObjectFromList

```
static Boolean getFileObjectFromList (const AEDesc *desc,
  AEDesc *resultDesc)
  {
    Boolean flFile = false;
    long numItems = 0;
    OSErr err = noErr;
    AEKeyword keyword;
    AEDesc tempDesc = {typeNull, NULL};

    if ((*desc).descriptorType != typeAEList)
      return (false);

    err = AECountItems (desc, &numItems);
    require_noerr (err, getFileObjectFromList_fail);

    if (numItems == 1) {
      err = AEGetNthDesc (desc, 1, typeWildCard,
        &keyword, &tempDesc);
      if (err == noErr) {
        flFile = getFileObject (&tempDesc, nil);

        if ((flFile) && (resultDesc != nil))
          AEDuplicateDesc (&tempDesc, resultDesc);
        AEDisposeDesc (&tempDesc);
        }
      }

    getFileObjectFromList_fail:
    return (flFile);
  }
```

First it makes sure that context is in fact an AEDescList. Then it counts the number of items in the list via AECountItems. If the number of items is one, then it gets the first item (AEGetNthDesc), then calls getFileObject to see if that first object is of typeAlias or can be coerced to typeAlias.

As with getFileObject, this function will return a copy of the file object in resultDesc as a typeAlias descriptor if the caller wishes. But for examineContext we don't want a copy, we just want to know if it's a file object or not.

Back to examineContext: if either getFileObject or getFileObjectFromList return true, then it calls

Your Data Isn't an Important Part of the Job

— It *IS* the Job.

VXA® FireWire

The High-Performance Tape Storage Solution For Apple Users

"I can stick my entire hard drive onto a tape 7 times —that's just plain cool!"

Other backup media, such as CD-RWs and DVD-RAMs, simply can't compare to VXA-1 tape technology when it comes to capacity, transfer rate and overall price. Fast, economical and easy-to-use, VXA FireWire offers:

Ultimate File Security

- 100% Data Restore and Interchange
- Plug-and-Play Connectivity

Sharable Media

- Multi-Gigabyte File Transport
- Portable, Hot-Pluggable

Real Time Digital Video

Cross-Platform Compatibility

- FireWire/IEEE1394/iLink Supported By All Major Manufacturers

LOWEST COST per MB Of ANY Technology!

Technology	Capacity in MBs	Transfer Rate in MB/sec	Price per MB
VXA Tape	**33000**	**3**	**$0.03**
Imation Travan	**10000**	**1**	**$0.05**
DVD RAM	9400	2.7	$0.05
CD RW	700	1.9	$0.50
Zip	250	1.2	$0.76

Call Exabyte sales at 1-800-774-7172 or visit us on the web at www.exabyte.com

Tape Storage By

VXA® Exabyte®

addCommandToMenu to add the Copy Path command to the contextual menu.

Adding a command

The contextual menu being built is passed to examineContext as an AEDescList. To add a command to the menu, addMenuCommand creates an AERecord and adds the command string and command ID. Then it adds the record to the end of commandList.

Listing 11: adding the Copy Path command to the contextual menu

addCommandToMenu

```
static OSStatus addCommandToMenu (AEDescList *commandList)
  {
    AERecord command = {typeNull, NULL};
    Str255 commandString = copyPathCommand;
    OSErr err = noErr;
    SInt32 commandId = 2000;

    err = AECreateList (NULL, 0, true, &command);
    require_noerr (err, addCommandToMenu_complete_fail);

    err = AEPutKeyPtr (&command, keyAEName, typeChar,
      &commandString [1], commandString [0]);
    require_noerr (err, addCommandToMenu_fail);

    err = AEPutKeyPtr (&command, 'cmcd',
      typeLongInteger, &commandId, sizeof (commandId));
    require_noerr (err, addCommandToMenu_fail);

    /*0 means add to end of list.*/
    err = AEPutDesc (commandList, 0, &command);

    addCommandToMenu_fail:
    AEDisposeDesc (&command);
    addCommandToMenu_complete_fail:
    return (err);
  }
```

The command is created via AECreateList. AEPutKeyPtr is used to add the text of the command and its command ID. (This sample doesn't actually use the command ID, since there's just one menu command—but if you have more than one command the command ID is quite useful.)

Finally the command record is added to the end of the command list via AEPutDesc.

After cleaning up, addCommandToMenu returns, then examineContext returns. Then we move on to implementing handleSelection.

handleSelection

This function, the second function required by the contextual menu interface, is called to run a contextual menu command. In this sample we run the Copy Path command. Our version of handleSelection looks like this:

Listing 12: handling the Copy Path command

handleSelection

```
static OSStatus handleSelection (void *pluginInstance,
  AEDesc *context, SInt32 commandId)
  {
    AEDesc aliasDesc = {typeNull, NULL};
    char path [MAXPATHLEN];
    OSErr err = noErr;

    if (!getAliasDesc (context, &aliasDesc))
      return (noErr);

    if (getPathStringFromAlias (&aliasDesc, path))
      writePathToClipboard (path);

    AEDisposeDesc (&aliasDesc);
    return (err);
  }
```

It takes a pointer to the plugin instance, a pointer to the context (the same context that examineContext gets, a description of the selected object(s)), and the command ID of the menu command that was chosen.

Just as in examineContext, we want to get a descriptor of typeAlias from context. So getAliasDesc is called to get a typeAlias descriptor. If it fails, the plugin does nothing.

Once it has the descriptor, it calls getPathStringFromAlias to get the path to the file object as a C string. Then it calls writePathToClipboard to write the path string to the clipboard. After cleaning up it returns.

getAliasDesc

Backing up... here's getAliasDesc:

Listing 13: getting a typeAlias descriptor

getAliasDesc

```
static Boolean getAliasDesc (const AEDesc *desc,
  AEDesc *resultDesc)
  {
    if (getFileObject (desc, resultDesc))
      return (true);
    return (getFileObjectFromList (desc, resultDesc));
  }
```

It calls getFileObject and getFileObjectFromList, which are the same functions we used in examineContext to determine if a file object was selected. The difference here is that the second parameter to these functions is not null, it's a pointer to a descriptor that should become a typeAlias descriptor—a file object.

getPathStringFromAlias

handleSelection then calls getPathStringFromAlias:

Listing 14: getting a path string from a typeAlias descriptor

getPathStringFromAlias

```
static Boolean getPathStringFromAlias (const AEDesc *desc,
  char *path)
  {
    FSRef fileRef;
    Size dataSize = AEGetDescDataSize (desc);
    AliasHandle aliasRef;
    Boolean flChanged = false;
    OSErr err = noErr;

    aliasRef = (AliasHandle) NewHandle (dataSize);
    if (aliasRef == nil)
      return (false);
```

The power of UNIX brought to the Mac

mac:ODBC
Open database connectivity for the Macintosh

Advanced SQL Editor
Powerful multi-platform and multi-DB SQL script editor

SQL Project
Multi-platform SQL development environment

Team Diagram
Dynamic diagramming and charting tool with CVS and database hooks

Data Architect
Multi-platform suite of database modelling and design tools: includes all the above software components

More Mac OS X products from theKompany.com coming soon

Take advantage of the power of UNIX with theKompany.com's suite of database software. Now also for Mac OS X.

TheKompany.com, a UNIX software and services company, now offers its powerful UNIX database connectivity tools for the Mac OS X computing platform.

With theKompany.com's suite of database tools, you can do just about anything with your SQL databases: connect to SQL databases with mac:ODBC, create and edit SQL scripts with the Advanced SQL Editor, create diagrams from CVS trees and databases with Team Diagram, develop powerful SQL-based projects with SQL Project and more.

Each of these tools is available separately, or as part of Data Architect – theKompany.com's premiere database modelling tool. All are offered with a unique open source license, so you can tweak the program code to your liking.

Prices for these great products start at just **$9.95** for individual packages and top out at **$49.95** for the complete Data Architect suite.

For more information about pricing and availability, to try a demo version, or to place your order online, visit us at **www.thekompany.com**.

theKompany.com

```
    err = AEGetDescData (desc, *aliasRef, dataSize);
    require_noerr (err, getFileObjectFromAlias_fail);

    err = FSResolveAlias (NULL, aliasRef,
      &fileRef, &flChanged);
    require_noerr (err, getFileObjectFromAlias_fail);

    err = FSRefMakePath (&fileRef, (UInt8*) path,
      MAXPATHLEN);

    getFileObjectFromAlias_fail:
    DisposeHandle ((Handle) aliasRef);
    return (err == noErr);
  }
```

It allocates a new AliasHandle via NewHandle, then gets its data by calling AEGetDescData to get it from the typeAlias descriptor.

Then it gets an FSRef from the AliasHandle by calling FSResolveAlias.

Then it gets a path string from the FSRef by calling FSRefMakePath. (Note: MAXPATHLEN is defined in sys/param.h, which is included by CopyPathPlugin.h.)

It returns after cleaning up.

writePathToClipboard

Finally handleSelection calls writePathToClipboard which writes the path string to the clipboard.

Listing 15: writing text to the clipboard

writePathToClipboard

```
static void writePathToClipboard (char *path)
  {
    ScrapRef scrap;
    ClearCurrentScrap ();
    GetCurrentScrap (&scrap);
    PutScrapFlavor (scrap, kScrapFlavorTypeText,
      kScrapFlavorMaskNone, strlen (path),
      (const void*) path);
  }
```

It clears the current scrap (clipboard), gets a reference to the current scrap, then writes the string as text to the scrap via PutScrapFlavor. Simple.

That's it for implementing handleSelection.

postMenuCleanup

This function is the third and last function required by the contextual menu interface. If the plugin had any cleanup to do after running the command, this would be the place to do it. But since there's no cleanup to do (in this sample), postMenuCleanup is a no-op:

Listing 16: cleaning up

postMenuCleanup

```
static void postMenuCleanup (void *pluginInstance)
  {
    /*This plugin has no cleanup to do.*/
  }
```

Testing and debugging plugins

After building your plugin, you'll find it in the build directory in the project directory. Copy it to ~/Library/Contextual Menu Items/. You may need to create the Contextual Menu Items folder if it doesn't already exist.

If you're replacing a plugin that's already installed, first trash the old version, then copy the new version into the Contextual Menu Items folder.

You'll need to quit and restart the Finder before it will see the new version of the plugin. What I do is put an AppleScript script in my dock that looks like this:

```
tell application "Finder"
  quit
end tell
```

OS X is just a bit unpredictable when it runs this script. Sometimes the Finder quits twice. Sometimes the Finder restarts, sometimes not. No harm is apparent. If the Finder doesn't restart, click the Mac smiley face icon in the Dock and the Finder will launch.

If you'd rather, you can log out then log back in instead, which also restarts the Finder.

To test this plugin, install it as above, then control-click (or right-click if you have a two-button mouse) on one file or folder in the Finder. You should see a Copy Path command in the menu that appears. Choose the command, then choose Show Clipboard from the Finder's Edit menu. The clipboard contents should be text, the path to the object you clicked on. Try doing a paste in another app just to make sure everything works as expected.

Tip: for debugging plugins I use printf calls. The output appears in Console.app, which you can find in the /Applications/Utilities/ directory.

Conclusion

As you've seen, implementing contextual menu plugins in OS X is not terribly difficult, it's just that the COM-compatible interface is not obvious. Given sample code and an understanding of how the COM interface works, you can concentrate on the code that's unique to your plugin.

Coming in part two

In the next article we'll go farther, we'll show how to build a submenu, how to send an update event to the Finder, how to handle selected text (in apps such as BBEdit and Internet Explorer), and how to call a Unix command from a menu command.

References

- Apple sample code: http://developer.apple.com/samplecode/Sample_Code/Human_Interface_Toolbox/SampleCMPlugIn.htm
- CoreFoundation Plugin Services: http://developer.apple.com/techpubs/macosx/CoreFoundation/PluginServices/Plug_in_ServicesConcTask/index.html
- Contextual Menu Workshop is a free framework for creating OS X contextual menus: http://homepage.mac.com/tkukiel/cmworkshop.html

by Tim Monroe

A Bug's Life

Retrieving Errors in QuickTime Applications

Introduction

Termites happen. So do errors in QuickTime-savvy applications — often for reasons other than mere sloppy programming. Network connections can fail in the middle of downloading a movie file or other data. System resources (memory, disk space, and so forth) can get depleted while an application runs. Components necessary for the playback of some media data might not be available on a particular machine. In short, lots of unpredictable occurrences can lead to the failure of QuickTime functions. How you deal with those failures is up to you. You might throw an exception, which (hopefully) is caught by an exception handler. Or you might just return an error code to your caller and expect it to handle the error gracefully. This is all part of the theory and practice of error handling, which is often the subject of heated debates among programmers. But before you even begin to *handle* an error, you first need to discover that it occurred in the first place. That's the subject of this article: how to determine that a QuickTime function has failed to do what you wanted it to do.

At first glance, this might seem like a fairly trivial topic. After all, many QuickTime functions return a result code that indicates the success or failure of the operation. But in fact things are not always that simple. For starters, not all QuickTime functions return a result code directly to the caller. Many of them, particularly Movie Toolbox calls, return a result code only indirectly, and we need to do a little work to retrieve that result code. We'll begin this article by looking at how to do that. Also, it's easy to misinterpret some of these result codes, so we'll investigate some of the pitfalls lurking here. Toward the end of the article, we'll take a look at a bug in our sample applications that I inadvertently added a few months ago.

Error-Reporting Functions

A large number of QuickTime functions return a result code directly to the caller as their function result. For instance, the EnterMovies function is declared essentially like this:

```
OSErr EnterMovies (void);
```

If a call to EnterMovies fails, QuickTime tells us so by returning a non-zero result code. The main reason that EnterMovies can fail is insufficient memory available for QuickTime to do the necessary initialization, so the result code is very likely to be memFullErr. No matter what the error here, however, our sample applications all quit pretty much immediately after they get one, first informing the user of the error. **Listing 1** shows a portion of our application start-up code on the Macintosh.

Listing 1: Initializing the Movie Toolbox (Macintosh)

main

```
myErr = EnterMovies();
if (myErr != noErr) {
  QTFrame_ShowWarning("\pCould not initialize QuickTime.
        Exiting.", myErr);
  ExitToShell();
}
```

And **Listing 2** shows the corresponding code in our Windows applications.

Listing 2: Initializing the Movie Toolbox (Windows)

WinMain

```
myErr = EnterMovies();
if (myErr != noErr) {
  MessageBox(NULL, "Could not initialize QuickTime.
        Exiting.", gAppName, MB_OK | MB_APPLMODAL);
  return(0);
}
```

But a significant number of QuickTime functions do not return an error code as their function result. A good example is StartMovie, which is declared like this:

```
void StartMovie (Movie theMovie);
```

As you can see, StartMovie returns no function result at all. Some other functions do return function results but they are not of type OSErr. An example here is GetMovieActive, which returns a result of type Boolean:

```
Boolean GetMovieActive (Movie theMovie);
```

Tim Monroe in a member of the QuickTime engineering team. You can contact him at monroe@apple.com. The views expressed here are not necessarily shared by his employer.

To handle cases like these, QuickTime provides a set of error-reporting functions. Let's see how these work.

Getting the Current Error

We can use the Movie Toolbox function GetMoviesError to retrieve the *current error value* (or *current error*), which is the result code of the most recently executed QuickTime function. GetMoviesError is declared like this:

```
OSErr GetMoviesError (void);
```

We can use GetMoviesError to get the result code for those functions that do not return one as their function result. (GetMoviesError also returns the result code for functions that *do* return an OSErr, but it's pretty much redundant in those cases.) Here's a typical use of GetMoviesError:

```
myTrack = NewMovieTrack(myMovie, myWidth, myHeight, 0);
myErr = GetMoviesError();
if (myErr != noErr)
  goto bail;
```

We could just as easily have checked to see whether myTrack is equal to NULL after the call to NewMovieTrack, but calling GetMoviesError gives us a result code that we can return to our caller, if so desired.

It's worth noting that GetMoviesError (and GetMoviesStickyError, which we'll consider in a moment) are global to an application and are not thread-specific. This means that an error that occurs in one thread can be reported to another thread. (Just something to keep in mind if you are writing multi-threaded applications.)

Getting the Sticky Error

QuickTime also maintains an error value called the *sticky error value* (or *sticky error*), which is the first non-zero result code of a Movie Toolbox function that was generated since the last time the sticky error was cleared. We retrieve the sticky error value by calling GetMoviesStickyError and we clear the sticky error by calling ClearMoviesStickyError. Here are the function prototypes:

```
OSErr GetMoviesStickyError (void);
void ClearMoviesStickyError (void);
```

When our application first starts up, the sticky error is 0. If all our Movie Toolbox function calls succeed, the sticky error remains set to 0. But as soon as any Movie Toolbox function encounters an error, the appropriate error value is copied into the sticky error value. We can call GetMoviesStickyError at any time to retrieve the sticky error value. This value does not change, even if subsequent Movie Toolbox calls fail, until we explicitly reset it to 0 by calling ClearMoviesStickyError.

The sticky error value is useful when we want to execute a series of Movie Toolbox functions but don't particularly want to check for errors after each Movie Toolbox call. **Listing 3** shows a situation in which GetMoviesStickyError might be used. The function VRObject_ImportVideoTrack copies a video track from one movie (the source) into a second movie (the destination).

Listing 3: Importing a video track from one movie into another

VRObject_ImportVideoTrack

```
OSErr VRObject_ImportVideoTrack (Movie theSrcMovie,
        Movie theDstMovie, Track *theImageTrack)
{
  Track       mySrcTrack = NULL;
  Media       mySrcMedia = NULL;
  Track       myDstTrack = NULL;
  Media       myDstMedia = NULL;
  Fixed       myWidth, myHeight;
  OSType      myType;
  OSErr       myErr = noErr;

  ClearMoviesStickyError();

  // get the first video track in the source movie
  mySrcTrack = GetMovieIndTrackType(theSrcMovie, 1,
        VideoMediaType, movieTrackMediaType);
  if (mySrcTrack == NULL)
    return(paramErr);

  // get the track's media and dimensions
  mySrcMedia = GetTrackMedia(mySrcTrack);
  GetTrackDimensions(mySrcTrack, &myWidth, &myHeight);

  // create a destination track
  myDstTrack = NewMovieTrack(theDstMovie, myWidth, myHeight,
        GetTrackVolume(mySrcTrack));

  // create a destination media
  GetMediaHandlerDescription(mySrcMedia, &myType, 0, 0);
  myDstMedia = NewTrackMedia(myDstTrack, myType,
        GetMediaTimeScale(mySrcMedia), 0, 0);

  // copy the entire track
  InsertTrackSegment(mySrcTrack, myDstTrack, 0,
        GetTrackDuration(mySrcTrack), 0);
  CopyTrackSettings(mySrcTrack, myDstTrack);
  SetTrackLayer(myDstTrack, GetTrackLayer(mySrcTrack));

  // an object video track should always be enabled
  SetTrackEnabled(myDstTrack, true);

  if (theImageTrack != NULL)
    *theImageTrack = myDstTrack;

  return(GetMoviesStickyError());
}
```

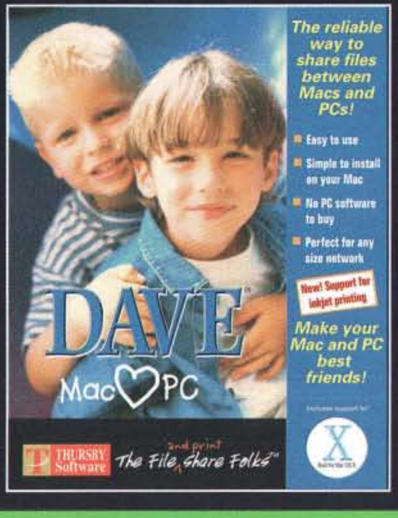
As you can see, we call ClearMoviesStickyError at the beginning of this function and then return to our caller the value returned by GetMoviesStickyError. The idea here is that our caller will care only about the first error we encounter while executing this function, which will of course be the sticky error (since we cleared the sticky error at the beginning).

Another case where we may want to access the sticky error is when we know or suspect that a QuickTime function will report an error, but we don't really care about that error. **Listing 4** defines a function, QTUtils_GetFrameCount, which returns the number of frames in a specified track. We use GetTrackNextInterestingTime to step through the track's samples.

Listing 4: Counting the frames in a track

QTUtils_GetFrameCount

```
long QTUtils_GetFrameCount (Track theTrack)
```


The reliable way to share files between Macs and PCs!
Easy to use
Simple to install on your Mac
No PC software to buy
Perfect for any size network
New! Support for inkjet printing
Make your Mac and PC best friends!
DAVE
Mac ♡ PC
THURSBY Software
The File, and print Share Folks™

DAVE® SAVED MY MAC

They were going to take our Macs away. Then I met DAVE.

DINOSAUR FOUND IN OFFICE

Manager Eaten Alive

"DAVE SAVED MY SANITY"

IT Analyst Timothy Hixon, Los Rios Community College System explained, "We didn't have time to learn another network operating system. The Macs needed to fit seamlessly and efficiently into our district's computer network, which used Windows NT file servers. DAVE was the answer."

SHARE FILES AND PRINTERS - Mac/PC, PC/Mac

"DAVE is, without any doubt, the best solution for integrating your Mac into a PC environment."

-Univers Macworld

★★★★★

ALL THE BUZZ about DAVE

"Essential Tool Gives Macs Hassle-Free Access to Windows Networks... this product is a must-have."

Shelly Brisbin
Macworld Magazine

Macworld
4 1/2

How did DAVE save your Mac?

Write us and tell us your story of how DAVE solved your problem.
The entries chosen for publication will be awarded a special Thursby attache'. Send entries to: DAVE-stories@thursby.com

www.thursby.com

THURSBY Software

The File, and print Share Folks™

```
{
  long            myCount = -1;
  short           myFlags;
  TimeValue       myTime = 0;
  OSErr           myErr = noErr;

  if (theTrack == NULL)
    goto bail;

  myErr = GetMoviesStickyError();

  // we want to begin with the first frame (sample) in the track
  myFlags = nextTimeMediaSample + nextTimeEdgeOK;

  while (myTime >= 0) {
    myCount++;

    // look for the next frame in the track; when there are no more frames,
    // myTime is set to -1, so we'll exit the while loop
    GetTrackNextInterestingTime(theTrack, myFlags, myTime,
        fixed1, &myTime, NULL);

    // after the first interesting time, don't include the time we're currently at
    myFlags = nextTimeStep;
  }

  if (myErr == noErr)
    ClearMoviesStickyError();

bail:
  return(myCount);
}
```

GetTrackNextInterestingTime returns, in the sixth parameter, the first time value it finds that satisfies the search criteria specified in the flags parameter. When it cannot find a time value that satisfies those criteria, it sets that parameter to –1. For all we know, it's possible that GetTrackNextInterestingTime also sets an error value; if so, we want to clear that value by calling ClearMoviesStickyError (but only if the sticky error on entry to our function was noErr).

Error Notification Functions

QuickTime provides the SetMoviesErrorProc function, which we can use to install an *error notification function* (or, more briefly, *error function*). An error notification function is called whenever QuickTime encounters a non-zero result code during the execution of a Movie Toolbox function. SetMoviesErrorProc is declared like this:

```
void SetMoviesErrorProc (MoviesErrorUPP errProc,
        long refcon);
```

The first parameter is a universal procedure pointer to our custom error notification function; the second parameter is a 4-byte reference constant that is passed to our error function when it is called. The error notification function is declared like this:
void MyMoviesErrorProc (OSErr theErr, long theRefcon);
The first parameter is the non-zero result code that was just encountered, and the second parameter is the reference constant we specified when we called SetMoviesErrorProc.

An error notification function is useful during application development or debugging, as they provide a single location where all errors are reported. This keeps us from having to put breakpoints all through our code as we track down problems.

Mysterious Errors

While we're on the topic of retrieving errors in QuickTime-savvy applications, it's worth discussing an issue that trips people up occasionally. This is the issue of mysterious QuickTime errors like –32766, which can occur when we execute some code like this:

```
OSErr    myErr = GraphicsExportSetDepth(myComponent, 32);
```

When this code is executed, then for certain graphics exporters, myErr is set to –32766. If we look in the file MacErrors.h, we won't find any such error. What's going on?

The explanation is surprisingly straightforward: GraphicsExportSetDepth and many other QuickTime functions that work with components return a function result of type ComponentResult, which is declared like this:

```
typedef long        ComponentResult;
```

On the other hand, the OSErr data type is declared like this:

```
typedef SInt16      OSErr;
```

When we try to fit a ComponentResult into an OSErr, we get only the low-order 16 bits, interpreted as a signed value. When the ComponentResult is noErr, this truncation is unproblematic. But several component errors use the full 32 bits of the long word, in which case the truncation will give us the mysterious errors described above. In particular, if a component does not support a particular action, then it will return the value badComponentSelector, which is defined as 0x80008002. Truncating 0x80008002 to a 16-bit signed quantity gives us –32766. That's what's happening with the call to GraphicsExportSetDepth we just considered: the particular component specified by the myComponent parameter does not support setting the export bit depth, in which case it returns badComponentSelector.

The lesson here is simple: pay attention to the data type of a function's return value and make sure you have enough space to hold that value. More specifically: don't use a variable of type OSErr to hold the return value of a component-related function whose return value is of type ComponentResult. But don't feel bad if you slip up occasionally. This mix-up is in fact so common that the file MacErrors.h contains some helpful comments:

```
/* ComponentError codes*/
enum {
  badComponentInstance= (long)0x80008001, /* when cast to an
OSErr this is -32767*/
  badComponentSelector= (long)0x80008002  /* when cast to an
OSErr this is -32766*/
};
```

A Framework Bug

Let's close this article by squashing a particularly nasty bug that I introduced into our sample applications a few months back, when we updated our Macintosh code to use Carbon events instead of "classic" events. (See "Event Horizon" in *MacTech*, May 2002.) Recall that we added a Carbon event loop timer to each open movie window, so that we can periodically task the movie controller (by calling MCIsPlayerEvent or MCIdle). Unfortunately, our existing application can crash — at least on Mac OS 9 — if we do something so simple as open a movie window and then later close it. That's not good.

Fixing the Bug

The problematic code turns out to be in the Macintosh version of the QTFrame_CreateMovieWindow function, shown in **Listing 5**. Here we create a new window and window object. Then we attach standard and custom Carbon event handlers to the window. Finally, we call InstallEventLoopTimer to attach a timer to the window.

Listing 5: Creating a movie window

QTFrame_CreateMovieWindow

```
WindowReference QTFrame_CreateMovieWindow (void)
{
  WindowReference   myWindow = NULL;

  // create a new window to display the movie in
  myWindow = NewCWindow(NULL, &gWindowRect, gWindowTitle,
        false, noGrowDocProc, (WindowPtr)-1L, true, 0);

  // create a new window object associated with the new window
  QTFrame_CreateWindowObject(myWindow);

#if USE_CARBON_EVENTS
{
  EventTypeSpec   myEventSpec[] = {
    {kEventClassKeyboard, kEventRawKeyDown},
    {kEventClassKeyboard, kEventRawKeyRepeat},
    {kEventClassKeyboard, kEventRawKeyUp},
    {kEventClassWindow, kEventWindowUpdate},
    {kEventClassWindow, kEventWindowDrawContent},
    {kEventClassWindow, kEventWindowActivated},
    {kEventClassWindow, kEventWindowDeactivated},
    {kEventClassWindow, kEventWindowHandleContentClick},
    {kEventClassWindow, kEventWindowClose}
  };

  // install Carbon event handlers for this window
  InstallStandardEventHandler
        (GetWindowEventTarget(myWindow));
  if (gWinEventHandlerUPP != NULL)
    InstallEventHandler(GetWindowEventTarget(myWindow),
        gWinEventHandlerUPP, GetEventTypeCount(myEventSpec),
        myEventSpec,
        QTFrame_GetWindowObjectFromWindow(myWindow), NULL);
}
  if (gWinTimerHandlerUPP != NULL)
    InstallEventLoopTimer(GetMainEventLoop(), 0,
              TicksToEventTime(kWNEMinimumSleep),
              gWinTimerHandlerUPP, myWindowObject,
              &(**myWindowObject).fTimerRef);
#endif

  return(myWindow);
}
```

It turns out that InstallEventLoopTimer can move memory, which might invalidate its last parameter, &(**myWindowObject).fTimerRef. If the window object indeed moves, then InstallEventLoopTimer will write the timer reference into the previous location of the window object. That's bad enough, but it gets worse when you realize that the window object, in its new memory location, now won't contain the timer reference returned by InstallEventLoopTimer. Rather, (**myWindowObject).fTimerRef will still be NULL. The event loop

Impact your bottom line while working with dedicated proponents of Apple technologies

With us, your only constant is success. Our custom solutions help our clients excel their business objectives and effectively target their audience. That's why we suggest that we run your technology while you play the winning shot.

Core Competencies
WebObjects development
Mac OS X application development
Cocoa application porting
Carbon porting
Mac OS product maintenance
Windows/Mac OS/Unix porting
Cross-platform development

Service Offerings
Offshore Project Development
On-Site Staff Augmentation
Off-Site Project Development
User Training

For More Information
visit http://www.avestacs.com
Email: mithu@avestacs.com

Avesta Computer Services, Ltd.
USA . EU . Asia

http://www.
scientific.com

Professional Software Developers
Looking for career opportunities?
Check out our website!
Nationwide Service
Employment Assistance
Resume Help
Marketability Assessment
Never a fee

Scientific Placement, Inc.
800-231-5920 800-757-9003 (Fax)
das@scientific.com

timer indeed gets installed, but we don't have a reference to it.

This in itself isn't a problem until we try to remove the event loop timer when the window is closed. Here's the code we use to do that:

```
if ((**myWindowObject).fTimerRef != NULL)
  RemoveEventLoopTimer((**myWindowObject).fTimerRef);
```

Since (**myWindowObject).fTimerRef is indeed NULL, RemoveEventLoopTimer isn't called and the timer continues firing even after the movie window has disappeared. **Listing 6** shows our event loop timer callback function.

Listing 6: Handling event loop timer callbacks

QTFrame_CarbonEventWindowTimer

```
PASCAL_RTN void QTFrame_CarbonEventWindowTimer
        (EventLoopTimerRef theTimer, void *theRefCon)
{
#pragma unused(theTimer)
  WindowObject myWindowObject = (WindowObject)theRefCon;

  // just pretend a null event has been received....
  if ((myWindowObject != NULL) &&
                ((**myWindowObject).fController != NULL))
    if (!gMenuIsTracking || gRunningUnderX)
      MCIdle((**myWindowObject).fController);
}
```

If the window object has been disposed of, then reading any of its fields (in this case, fController) will likely result in a segmentation fault or other error.

This is a classic case of using a *dangling pointer*, the address of a block of memory whose contents have moved. You can get the full details on this type of problem in the book *Inside Macintosh: Memory* (which I am presently chagrined to admit I myself wrote a decade ago). There are several solutions to this type of problem. A standard solution is to lock the window object before calling InstallEventLoopTimer and then unlock it afterwards:

```
HLock((Handle)myWindowObject);
if (gWinTimerHandlerUPP != NULL)
  InstallEventLoopTimer(GetMainEventLoop(), 0,
                TicksToEventTime(kWNEMinimumSleep),
                gWinTimerHandlerUPP, myWindowObject,
                &(**myWindowObject).fTimerRef);
HUnlock((Handle)myWindowObject);
```

Or, even more simply, we can just use a temporary variable to hold the timer reference:

```
EventLoopTimerRef     myTimerRef;

if (gWinTimerHandlerUPP != NULL)
  InstallEventLoopTimer(GetMainEventLoop(), 0,
                TicksToEventTime(kWNEMinimumSleep),
                gWinTimerHandlerUPP, myWindowObject,
                &myTimerRef);
(**myWindowObject).fTimerRef = myTimerRef;
```

Adding Some More Protections

Let's take this opportunity to tinker with the Carbon event loop timer callback function QTFrame_CarbonEventWindowTimer (**Listing 6**, above). First of all, we should add a check at the top of the function to make sure we got a non-NULL window object:

```
if (myWindowObject == NULL)
  return;
```

And we should make sure that we are passed the same timer reference we are storing in the window object:

```
if ((**myWindowObject).fTimerRef != theTimer)
  return;
```

More importantly, I want to change the call to MCIdle into a call to MCIsPlayerEvent. We can achieve this end by building a null event and passing it to our framework function QTFrame_HandleEvent, as shown in **Listing 7**.

Listing 7: Handling event loop timer callbacks (revised)

QTFrame_CarbonEventWindowTimer

```
PASCAL_RTN void QTFrame_CarbonEventWindowTimer
        (EventLoopTimerRef theTimer, void *theRefCon)
{
  WindowObject myWindowObject = (WindowObject)theRefCon;

  if (myWindowObject == NULL)
    return;

  // sanity check: make sure it's our timer
  if ((**myWindowObject).fTimerRef != theTimer)
    return;

  // just issue a null event to our event-handling routine....
  if (!gMenuIsTracking || gRunningUnderX) {
    EventRecordmyEvent;

    myEvent.what = nullEvent;
    myEvent.message = 0;
    myEvent.modifiers = 0;
    myEvent.when = EventTimeToTicks(GetCurrentEventTime());
    QTFrame_HandleEvent(&myEvent);
  }
}
```

I prefer this revised approach to tasking our movie controllers because it routes null events through our existing event-handling routine QTFrame_HandleEvent. This in turn will make it easier to modify our code to handle movies that need to be tasked but which don't yet have a movie controller attached to them. In the next article, we'll see how this can happen.

Conclusion

Of the four new QuickTime functions we've encountered in this article (GetMoviesError, GetMoviesStickyError, ClearMoviesStickyError, and SetMoviesErrorProc), we're most likely to want to use GetMoviesError in our daily programming, as it provides our only means of retrieving the result codes for a large number of QuickTime functions. I generally find the sticky error less useful, but there are times we might want to take a look at it. The error procedure is, to my knowledge, largely unused. I can, however, imagine that a clever programmer could find some useful applications for it, so it's good to at least know it exists.

MacDirectory Magazine

Creative designers, writers, musicians, business leaders and our expert technology team offer their own personal interpretation of things that only the Macintosh system can deliver. Featuring over 240 pages of reviews, product news, insights, trends and the largest Macintosh buyers' guide - including over 5,000 products and services for your mac.

Interviews

Tapping into the world of celebrities and their macs, only MacDirectory offers in depth interviews. Get a close and personal view on Steve Jobs, Tom Cruise, Claudia Schiffer, Madonna, George Lucas and other leaders in the Mac community.

Culture

MacDirectory takes you to the wildest corners of the globe and uncovers many uses of Macintosh computers. Travel to Japan, Australia, Germany, India, Brazil, Russia and learn more about Apple's cultural impact around the globe.

Subscribe

Subscribe online for faster delivery: **www.macdirectory.com/sub.html** Subscription rates follows: 1 year, $32.00 for 4 issues & 2 years, $60.00 for 8 issues. Or you may also mail a check or money order to: MacDirectory Subscription Dept. 150 West 25th St NY NY 10001. Include your phone, email and mailing address.

By Andrew C. Stone

The Philosophy of Cocoa: Small is Beautiful and Lazy is Good

I believe a programming renaissance is upon us—and Cocoa, Apple's high level object-oriented framework is at the heart of it. By wrapping complexity inside easy-to-use objects, Cocoa frees application developers from the burden of the minutiae that so often drives developers crazy. Instead, they can focus on what's special about their applications, and in a few lines of code, create a complete OS X application that seamlessly interoperates with all other OS X applications. And, for very little additional effort, they get AppleScriptability as well.

I've been living and breathing Cocoa and its various previous incarnations for 14 years now, and have noticed that my applications are getting more features with smaller amounts of code each year. This article will explore some of the truisms and gems hard earned by hanging in the trenches lo these many years.

SMALL IS BEAUTIFUL

It's no coincidence that we use the term "architecture" for the overarching structure of an application. I received my baccalaureate in classical Architecture in the '70's when the visionaries of the time were rebelling against the huge concrete boxes that crushed the human scale and spirit. ***E. F. Schumacher, in Small is Beautiful — A Study of Economics as if People Mattered:***

"What I wish to emphasise is the duality of the human requirement when it comes to the question of size: there is no single answer. For his different purposes man needs many different structures, both small ones and large ones, some exclusive and some comprehensive. Yet people find it most difficult to keep two seemingly opposite necessities of truth in their minds at the same time. They always tend to clamour for a final solution, as if in actual life there could ever be a final solution other than death. For constructive work, the principal task is always the restoration of some kind of balance. Today, we suffer from an almost universal idolatry of giantism. It is therefore necessary to insist on the virtues of smallness - where this applies. ..." (p. 54)

I believe this thinking still rings true 30 years later in cyber-architecture.

From the programming classic The Mythical Man Month by Frederick P. Brooks, Jr., we learned that the more programmers you throw at a project, the less likely the project will ever be finished! From this follows that a project should have one central architect, with rather fascist control over feature set and implementation, especially if you want it to ship in a timely fashion. Cocoa gives you the tools needed to build full-featured, world-class applications with just a handful of programmers. For best effect, these programmers should be lazy...

Lazy is Good

Laziness is a virtue, believe it or not! I often describe my style of a computer scientist as someone who is so lazy that they'll spend days writing software to save a minute each time the task is performed from then on. There are two forms of laziness that Cocoa embraces: lazy loading of objects and just plain lazy programming.

Bundle it Up

Lazy loading lets full-featured applications like Stone Design's Create®, a three-in-one illustration, page layout and web authoring app, launch in just a few seconds. Compare that to a legacy Carbon application with half the features which takes minutes to launch! By using dynamically loaded bundles, you do not use memory or resources until the end user actually needs that particular feature and its related

Andrew Stone, CEO of Stone Design, www.stone.com, has been the principal architect of several solar houses and over a dozen Cocoa applications shipping for Mac OS X.

resources. Moreover, you can update and distribute just the tiny bundle instead of the whole application should, heaven forbid, a bug be found!

To use dynamically loaded bundles, you need to be able to compile your application without actually referencing the loadable object directly. We do this runtime magic by only referring to the dynamically loaded class, the principal class of the bundle, by its name as a string.

Typically, the types of objects that do well being loaded dynamically are the numerous special editors and interfaces in a program, such as an arrow or pattern editor and the classes it needs to provide the interface. The following conditions make up a good candidate for a loadable bundle:

- Has resources that are not always used each session
- Doesn't contain core data model classes (these should be linked)

A typical example might have an NSWindowController subclass and perhaps some custom views and images in the bundle. We load this type of bundle by having it respond to the class method "+sharedInstance", since you usually only need one of these objects per application:

```
- (void)loadAUniqueInterfaceObjectAction:(id)sender {
  // we only use the name of the bundle, not its class
  // which would cause an undefined symbol when linking:
    [[[NSApp delegate]
sharedInstanceOfClassName:@"MyUniqueController"]
showWindow:self];
}
```

But, with the introduction of sheets, two or more documents may want to load the same bundle (for example, a custom zoom sheet) at the same time. In this case, you'll want a unique instance, which will be released after use. These principal classes of bundles need only respond to -(id)init, which all objects do anyway since they inherit from NSObject which implements -(id)init;

```
- (void)loadAPerDocumentInterfaceObjectAction:(id)sender {
  [[[NSApp delegate]
instanceOfClassName:@"MyPerDocumentController"]
showWindow:self];
}
```

And, because we are lazy and more importantly, good programmers, we filter both of these methods through one factored method, -(id)instanceOfClassName:(NSString *)name shared:(BOOL)shared like this:

```
- (id)sharedInstanceOfClassName:(NSString *)name
{
    return [self instanceOfClassName:name shared:YES];
}

- (id)instanceOfClassName:(NSString *)name {
    return [self instanceOfClassName:name shared:NO];
}
```

And here's the non-linked bundle loading code for both of them, which we place for convenience in the globally available [NSApp delegate] class:

```
- (id)instanceOfClassName:(NSString *)name
shared:(BOOL)shared
{
     id obj = nil;
     NSString *path = [[NSBundle mainBundle]
pathForResource:name ofType:@"bundle"];
     if (path) {
    // we found the bundle, now load it:
          NSBundle *b = [[NSBundle allocWithZone:NULL]
initWithPath:path];
    // if it loads, see if it has a valid principalClass - this is set in PB's target
inspector
          if ((b != nil) && ([b principalClass] !=NULL)) {
    // here is the only difference between a single shared instance
    // and a new one every time:
               if (shared) obj = [[b principalClass]
sharedInstance];
               else obj = [[[b principalClass]
allocWithZone:NULL] init];
          } else {
  //This is for debugging in case it can't be loaded:
  NSLog(@"Can't Load %@!\n", path);
          }
     } else NSLog(@"Couldn't find %@ bundle!\n",name);
     return obj;
}
```

Felt Tip Software presents

Sound Studio 2.0

Built for Mac OS X

With Sound Studio 2.0, you can:

Digitize cassette tapes and vinyl records to make CDs
Record live performances and interviews
Record your favorite radio programs with Timer Recording
Edit out dead air, glitches, and mistakes in recording
Apply any of 19 effects including Reverb and Chorus
Create new sounds and loops

Sound Studio is an audio recording and editing application built for Mac OS X. It can also run on Mac OS versions 8.1 and up. Some systems may need a USB audio input device to record.

Only $49.99
Download 14-day trial version at
http://www.felttip.com/ss/

felt tip software
807 Keely Place, Philadelphia, PA 19128, USA
www.felttip.com

Bridge the communication gap, make the right connection — with innovative solutions at

DEV DEPOT℠

www.devdepot.com

Griffin Pro Speaker Breakout Cable

$23.95

Connect standard speakers to your Mac pro speaker socket!

SMARTLINK USB File Transfer Cable

$39.95

Transfer files over USB, between Mac, PC's, or across platforms!

PO Box 5200 • Westlake Village, CA • 91359-5200 • Voice: 877-DEPOT-NOW (877-337-6866)

DVIator DVI to ADC Adapter

Connect flat panel monitors to your DVI video source!
(yes we have PCI DVI video cards too!)

$147^{95}

Belkin USB VideoBus II for Mac

Fast and easy video capture over USB!

$79^{95}

Griffin iMic

Sound in and out over USB!

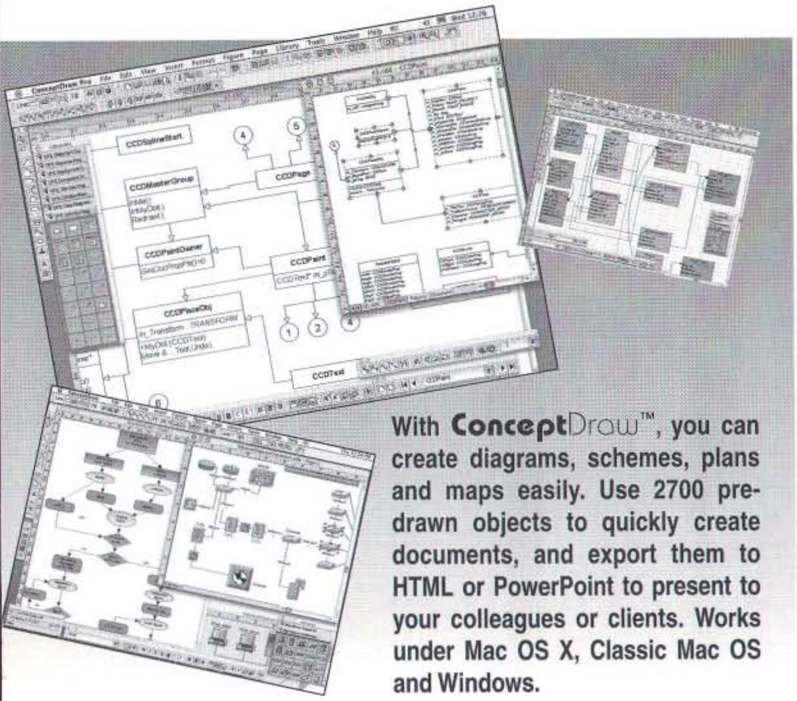
$32^{95}

Don't have the right port? Maybe you just need the right cable!

Outside US/Canada: 805/494-9797 • Fax: 805/494-9798 • E-mail: orders@devdepot.com

ConceptDraw™ PROFESSIONAL

Cross-platform diagramming and drawing tool



With ConceptDraw™, you can create diagrams, schemes, plans and maps easily. Use 2700 pre-drawn objects to quickly create documents, and export them to HTML or PowerPoint to present to your colleagues or clients. Works under Mac OS X, Classic Mac OS and Windows.

Use ConceptDraw Pro for:

- **UML**
- **Network Diagrams**
- **Flowcharts**
- **Web Site Schemes**
- **Interface Modelling**
- **Process Flow**

Download demo or order online at:
www.conceptdraw.com/professional
e-mail: sales@conceptdraw.com

Available @ DEPOT 877-DEPOT-NOW or 805-494-9797 fax 805-494-9798 http://www.devdepot.com orders@devdepot.com

Code Lazily

Lazy programming means "Use the 'Kit, Luke!" Every standard data structure and a complete set of API's are already available to you, so there is rarely a need to reinvent your own. Therefore, this lazy programming axiom has a corollary:

If it's hard to do or understand, it's wrong

By this I mean any coding solution that involves convoluted logic or going beneath the API (using undocumented methods) is probably not the right approach. Taking the time to understand what's already offered to you is well worth the effort, because Cocoa, and its underlying frameworks, Foundation and AppKit, have evolved over 16 years to provide the basic building blocks of an object oriented solution. Many times when I'm adding a new feature, I'll try one brute force approach, notice how cumbersome it is, re-read the AppKit or Foundation API and find a much a better solution involving much less code.

For example, I recently added a "Clone Client" feature to TimeEqualsMoney™ - take the current document's data, remove the individual time entries, create a new document with all the same settings except no time entries. It was six lines of code and it worked the first time:

```
- (void)cloneDocumentAction:(id)sender {
  // make a new untitled - have it read our document:
    NSMutableDictionary *doc = [self
workDocumentDictionary];
    MyDocument *newDoc = [[NSDocumentController
sharedDocumentController]
openUntitledDocumentOfType:DocumentType display:NO];

    [doc removeObjectForKey:WorkKey];
    [newDoc loadDataRepresentation:[[doc description]
dataUsingEncoding:NSASCIIStringEncoding]
ofType:DocumentType];
    [newDoc makeWindowControllers];
    [[[newDoc windowControllers] objectAtIndex:0]
showWindow:self];
}
```

Instead of hiring programmers, why not let the entire Cocoa team at Apple be your engineers? When you use the Kit and Apple puts in new functionality and bug fixes, your application automatically gains these features and fixes. Your efforts should be focused on creating a mapping between the real world problems you are solving and the objects that represent them. Which brings me to my next point:

Put The Code Where It Belongs

One of the biggest challenges facing newcomers to Object Oriented Programming is placing code in the right object. Because many of us grew up with "procedural" languages like Basic, Pascal, and C, and because old habits die hard, we need to let go of trying to tell things what do to do, and instead, let them figure it out for themselves. Let's say we have a document which is a list of pages, which contains a list of graphics, which are simple graphics or groups, which contain a list of graphics, which are graphics or groups which contain, etc... And let's say we want to set the "isVisible" state of all the graphics in the document.

The procedural approach would be to assume absolute knowledge over this hierarchy, and you'd blithely code something like this:

```
@implementation MyDocument

// please don't do this!
- (void)setAllObjectsVisible:(BOOL)isVisible {
     unsigned int i, pageCount = [_pages count];
     for (i = 0; i < pageCount; i++) {
          Page *p = [_pages objectAtIndex:i];
          NSArray *graphics = [p graphics];
          unsigned int j, graphicsCount = [graphics count];
          for (j = 0; j < graphicsCount; j++) {
               Graphic *g = [graphics objectAtIndex:j];
               if ([g isKindOfClass:[Group class]) {
                    NSArray *groupGraphics = [g graphics];
                    unsigned k,groupGraphicsCount =
[groupGraphics count];
                    for (k = 0; k < groupGraphicsCount; k++)
{
   // since this only recurses one level, this code is wrong as
   // well as very hard to read and maintain!!!
   Graphic *groupedGraphic = [groupGraphics
objectAtIndex:k];
   [k setVisible:isVisible];
                    }
               else [g setVisible:isVisible];
          }
     }
}

// the OO way:

@implementation MyDocument
- (void)setAllObjectsVisible:(BOOL)isVisible {
       [_pages
makeObjectsPerformSelector:@selector(setAllObjectsVisible:)
withObject:(id)isVisible];
}
...

@implementation Page
- (void)setAllObjectsVisible:(BOOL)isVisible {
       [_graphics
makeObjectsPerformSelector:@selector(setVisible:)
withObject:(id)isVisible];
}
....


// groups need to recurse down the hierarchy until individual graphics
// are found...

@implementation Group

- (void)setVisible:(BOOL)isVisible {
     [_graphics
makeObjectsPerformSelector:@selector(setVisible:)
withObject:(id)isVisible];
}


@implementation Graphic
- (void)setVisible:(BOOL)isVisible {
  // only do work if you absolutely have to - remember LAZY!
     if (_isVisible != isVisible) {
   // you'd probably do undo manager stuff here
   _isVisible = isVisible;
   // alert page we need to be redrawn
   [self tellMyPageToInvalidateMyBounds];
     }
}
...
```

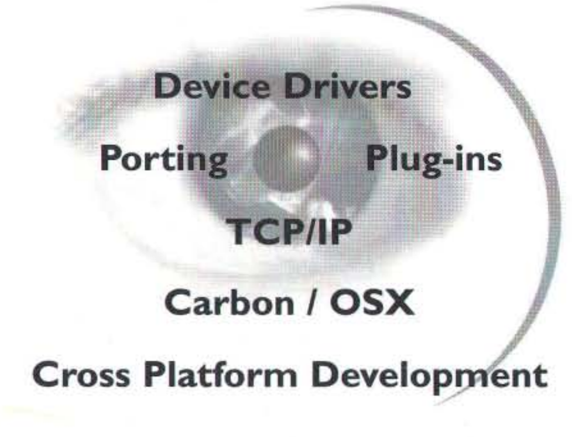
Conclusion

The more you understand object oriented programming and Cocoa, the smaller and more reusable your applications will become. And they will load with lightning speed! But more importantly, you'll have less code to maintain which means less bugs, less headaches and more time to enjoy life.

high quality - competitive rates - 16 years experience - award winning

reliable - high quality - competitive rates - efficient - award winning - qa services - reputable

Full Spectrum Software
Development & Testing



Device Drivers
Porting **Plug-ins**
TCP/IP
Carbon / OSX
Cross Platform Development

One Bridge Street
Newton, MA 02458

617-965-0029
www.FullSpectrumSoftware.com

reliable - high quality - competitive rates - efficient - award winning - qa services - reputable

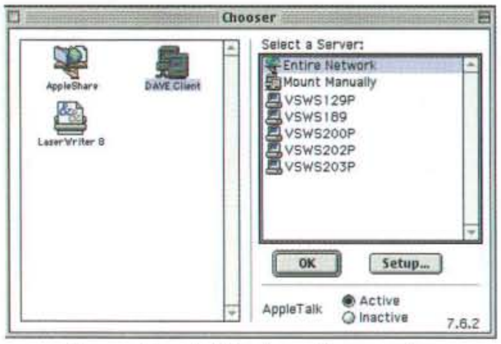
competitive rates - 16 years experience - award winning - high quality

By Michael R. Harvey

DAVE

Dave, from Thursby Software, is the gold standard when it comes to cross platform file and printer sharing. It is the best seller utility of it's kind, and shows every sign of remaining so. Features added to the latest version include support for Unicode file names, the ability to access files larger than 2 GB, and the availability of the program documentation via the Help Center, among others.

Installation of Dave is a breeze. In OS 8 and 9, run the installer, then restart. In OS X, not even a restart is needed. Dave is ready to go immediately.



Connecting to Windows shares using Dave in OS X, and OS 8 and 9.

Configuration of Dave is simplicity itself. After installation, simply follow the instructions in the Setup Assistant. The assistant will query you for the information it needs to establish a connection to the Windows servers on your network, like WINS server address and Workgroup name. You will also have the option during setup to share out a folder from your Mac to the Windows machines on your network. This can be done now, or skipped over. This setting, as well as all the others, can be changed at any time by going back through the Setup Assistant again. Running the Assistant starts you over at the beginning, but all your previous information will be there. You can easily make changes only to the areas you need, skipping past the parts that do not require updating.

Accessing Windows servers via Dave is now just like accessing any other network volume. The specifics of how depend on which operating system you are running.

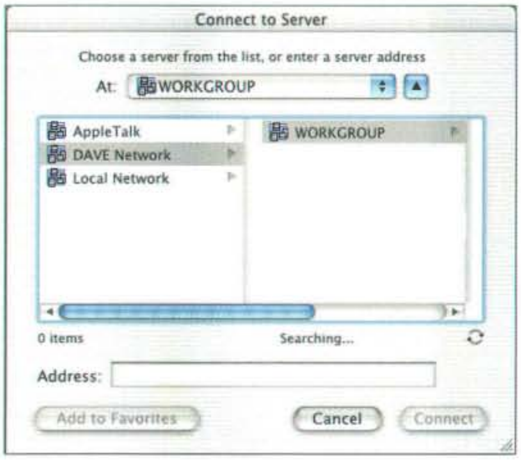
In OS X, accessing Windows servers is done via the Connect to Server window. The behavior is identical to mounting a sever via AppleTalk. From the Go menu, select Connect to Server. Dave Network appears along with AppleTalk and Local Network. Click on it, and you will get a listing of domains and workgroups. Choose the one you want to access, and a listing of available shares for that domain will pop up. From there, just double click on the share you wish to access, and logon as usual.

In Mac OS 8 and 9, you access those heretofore unreachable Windows shares via the Chooser. Once open, a Chooser extension named Dave Client is listed along with the usual selections, like AppleShare. Clicking on it will let you browse the entire network, mount a share manually, or select from the available list of Windows boxes on your network. Double click on the share you wish to access, enter your logon and password, as you would normally, and you are in.

Thursby provides some nice documentation to get you up and running. A printed Quick Start Guide will have you connected to your Windows network in minutes. For both OS X, and 8 and 9, extensive material is accessible via the Help Center, found in the Help menu. The Help Center data is well laid out, and finding answers is fairly easy. Some of the material to be had includes detailed set up instructions, and answers to common questions. A troubleshooting guide is key among the available material, providing extensive information on possible errors that you may encounter, and likely solutions.

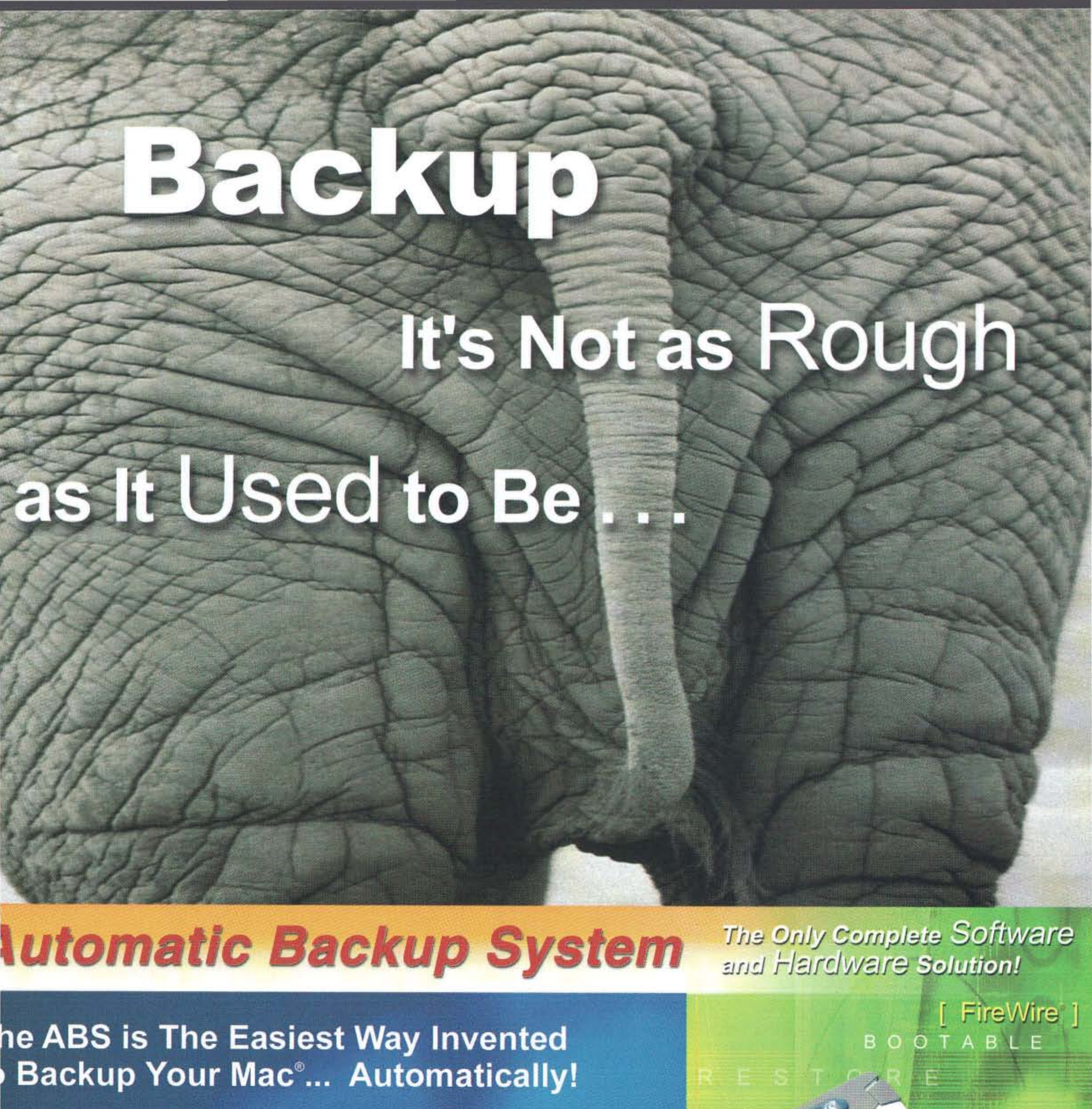
There have been some problems with the version 3.1 release of Dave. Thursby has been working to resolve these issues. In early May, they released version 3.1.1 to address some of these issues, as well as add features. Check the Read Me file for all the gory details. Thursby also maintains an up to date FAQ on it's website, that you can link to from the Help Center, that answers many of the questions you may have regarding installation, networking, or any other problems.

Another great resource for finding solutions to problems with Dave is the website, MacWindows. It deals with all issues relating to Mac and Windows integration, and is a great resource for troubleshooting not only Dave, but all aspects of cross platform networking.



Thursby Software sells a range of products for connecting Macs and Windows PC's together in environments ranging from the home office to the corporate LAN. They even offer a product, MacNFS, available to make connecting to Unix servers a simple process. If you plan on cross platform networking in OS X, you'll need Dave. Dave has an MSRP of $149 for a single license. Thursby also offers an Annual Update program. This program will give you all updates that are released within the twelve months of the contract. Pricing for the Annual Update program starts at $49.95 for a single license, up to $900 for fifty licenses.

http://www.thursby.com/
http://www.macwindows.com/



utomatic Backup System

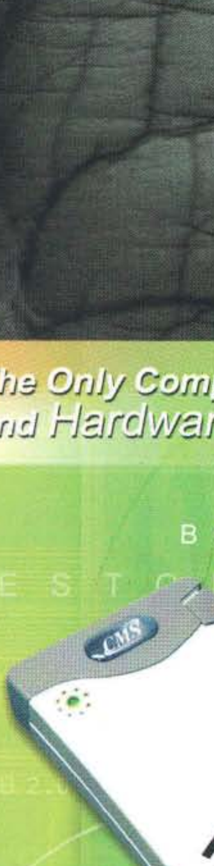
The Only Complete Software and Hardware Solution!

he ABS is The Easiest Way Invented
Backup Your Mac®... Automatically!

ackup, store & share files between iMacs™, iBooks™,
ower Macs™, & PowerBooks™, including video,
gital photography, graphics, & MP3s up to **60GB**!

r More Information Call **800 • 327 • 5773** or
ick www.**cmsproducts.com**/mt1

[FireWire]
BOOTABLE
RESTORE
SECURITY
USB 2.0



[High Speed]
RELIABILITY

CMS PERIPHERALS

Now Get 15% Off All Online Orders!

Mac

WEBOBJECTS DEVELOPMENT

By Tom Woteki

Reusable WebObjects Components

INTRODUCTION

One of the more interesting and powerful features of WebObjects is the ability to construct reusable components. However neither the documentation nor any of the available books on WebObjects cover this topic very well. The best discussions are in *Professional WebObjects 5.0 with Java* (DeMan, et. al. 2001), and the Windows and version 4.5-oriented *WebObjects Web Application Construction Kit* (Ruzek 2001). Both of these treatments, though instructive, leave a lot to the reader. The common example that both books cover is how to construct a standard template for an application.

This article illustrates how to build reusable components using two examples drawn from a WebObjects application I wrote to maintain records of a wine collection. It also illustrates creating custom component bindings and communications between components using key-value coding. The example components have wide applicability to the maintenance of reference tables and are easily generalized to maintaining any single attribute of a database table.

This article assumes you are familiar with at least the basics of building a WebObjects database application including using interface widgets, forms, actions, data models and display groups.

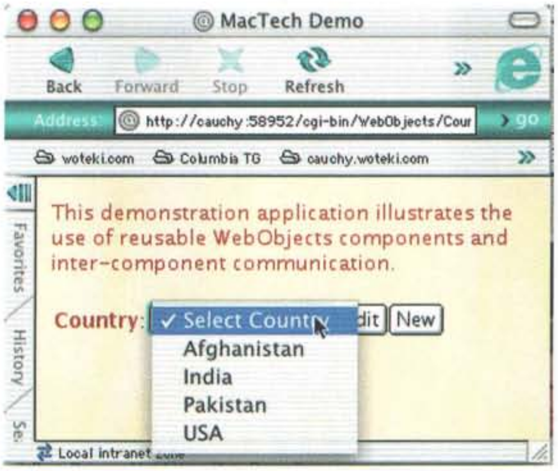
THE EXAMPLE PROBLEM: MAINTAINING A REFERENCE TABLE

A common type of table in a database application is a reference table, that is, a table consisting of two columns: a primary key and the attribute of interest. An example is a table of country names. In my application there are reference tables for wine producing countries, wine regions, wine names and so on. Given such reference tables, an obvious need is to maintain the tables, either to add new records or to update existing records. Because my application uses multiple reference tables, and noting the obvious generality of the situation, I decided to develop some reusable components for maintaining any reference table. We'll see how to use these components in an example application to maintain a reference table of country names.

HIGH-LEVEL DESIGN

The approach I took was to develop two components, each designed to be embedded in a form within a parent component. One component, the "selector", is used either to select an existing record to edit or to initiate insertion of a new record. The other, the "editor", is used to actually edit the selected record or create the new one. The two components communicate with each other through key-value coding whereby the selector passes the user's intent and other information to the editor. We use bindings to designate the specific table and attribute the components should edit.

Figures 1 and 2 show the respective finished products, each embedded in parent components. The selector component in Figure 1 consists only of the WOPopUpButton and the two form buttons "Edit" and "New". The former button initiates an update of a selected record, the latter initiates insertion of a new record. The remaining aspects of the interface, such as the prompt string, pertain to parent components such as the page itself. Similarly, the editor component in Figure 2 consists only of a WOTextField and two form buttons. The separate implementation of the components, apart from communications via key-value coding, and their separation from aspects of the surrounding interface, including the form they are embedded in, maximizes their reusability and adaptability to different interface designs.



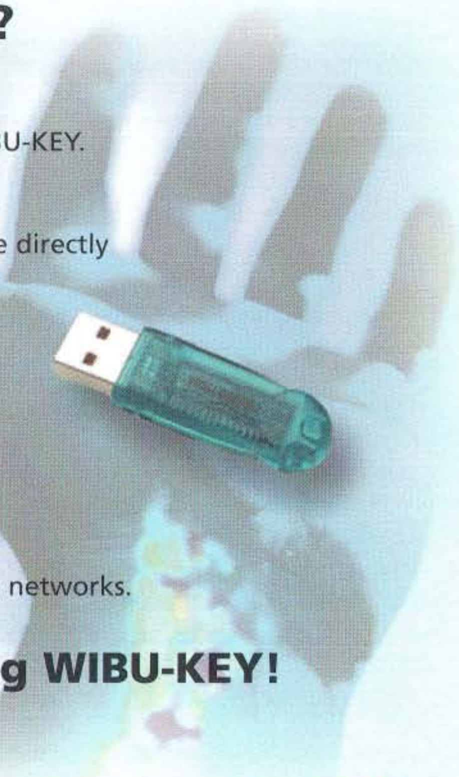
This demonstration application illustrates the use of reusable WebObjects components and inter-component communication.

Country: ✓ Select Country | Afghanistan | India | Pakistan | USA — Edit New

Figure 1: The finished selector component in action

In addition to being a Macintosh and WebObjects hobbyist developer, **Tom Woteki, aka Dr. Wo**, is a vice president at TRW Systems. He can be reached at drwo@woteki.com

The Key is in Your Hands!



Does your dongle do all this?

Software Goes Online
Electronic Software Distribution – safely protected by WIBU-KEY.

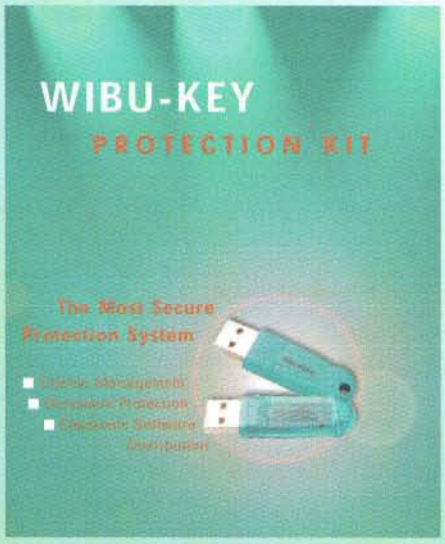
Web Remote Programming
Reprogram the WIBU-BOX hardware at the customer's site directly via the Internet.

Web Authentication
Secure authentication via a two-way-encryption.

Pay-Per-Use
Usage dependent accounting.

Mac OS 9 & X
WIBU-KEY supports Mac OS, Windows and heterogeneous networks.

▶ Yes? Then you are already using WIBU-KEY!



▶ No? Then order your Test Kit at 1-800-986-6578

You will find more information at
www.griftech.com

GRIFFIN TECHNOLOGIES

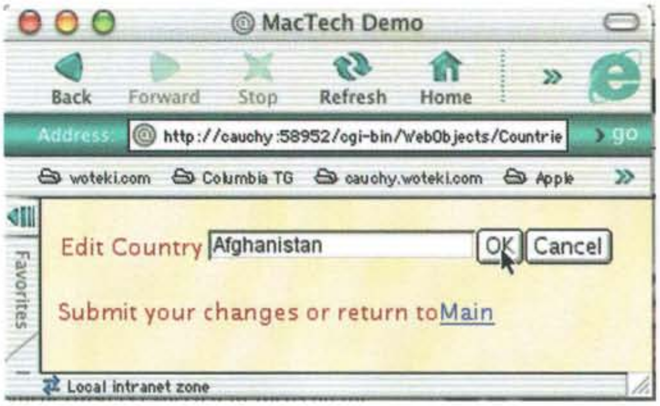
USA, Canada: Griffin Technologies, LLC
phone: (785) 832-2070 · fax: (785) 832-8787
email: sales@griftech.com · www.griftech.com

WIBU SYSTEMS

WIBU-SYSTEMS AG
76137 Karlsruhe, Germany
WIBU-SYSTEMS USA, Inc.
Seattle, WA 98101
email: info@wibu.com

www.wibu.com

Test Kits also available at:
Argentina info@safield-group.com, **Australia** simone.eckerle@wibu.com, **Belgium** wibu@impakt.be, **Croatia** aries@aries.hr, **Denmark** ivha@danbit.dk, **Finland** finebyte@finebyte.com, **France** info@neol.fr, **Hungary** info@mrsoft.hu, **Japan** info@suncarla.co.jp, **Jordan, Lebanon** starsoft@cyberia.net.lb, **Korea** dhkimm@wibu.co.kr, **Luxembourg** wibu@impakt.be, **Netherlands** wibu@impakt.be,**Portugal** dubit@dubit.pt, **Syria** starsoft@cyberia.net.lb, **Thailand** preecha@dptf-th.com,**United Kingdom** info@codework.com, **USA** sales@griftech.com



Edit Country Afghanistan

Submit your changes or return toMain

Figure 2: The finished editor component in action

DETAILED DESIGN

Now let's consider details of the preceding design. First of all, let's name the components we're going to build WOObjectSelector and WOObjectEditor. Viewing Figure 1 it should be clear that we must at least provide WOObjectSelector the list it should display in its WOPopUpButton menu and the page(s) to return when either its "Edit" or "New" buttons are clicked. In this design both buttons will return the same page. We might also want to provide the component a "no selection" string to display when there is no selection in the pop-up. Since we are editing a reference table, you can anticipate that we will bind a WODisplayGroup to the pop-up's display list. And you might anticipate that the return page for the buttons should be some page that encloses WOObjectEditor as a child component. Before we discuss these details, including how to provide WOObjectSelector the information it needs, let's consider what WOObjectEditor needs to do its job.

Viewing Figure 2, we can see that we need to provide the editor at least the data for the attribute of the record we are editing (in our example, the name of a country) plus the return page for the form buttons. Even more is needed, however. First of all it would be helpful to know if we are updating an existing record or inserting a new one. Second, if updating we'll need the record itself, not just the data for the attribute of interest. In order to achieve reusability, we'll provide not only the record, but the key for the attribute as well, so that we can use the powerful generality of key-value coding to update the value of any attribute. Finally, in case of inserting a new record, we'll need to create the record and insert it into the default editing context, so we'll need to provide WOObjectEditor the name of the entity to create as well as the key for the pertinent attribute.

Which component will provide WOObjectEditor the information it needs and how? WOObjectSelector knows whether the user is updating or inserting by means of the button the user clicks. If updating, it also knows the record the user is updating, namely the one corresponding to the item selected in the display list. So, WOObjectSelector is the logical provider of this information. It will do so by invoking the key-value coding method takeValueForKey (which is inherited by any class that extends WOComponent) on the parent page that encloses WOObjectEditor. It will invoke the method for each of a series of keys corresponding to the data needed by WOObjectEditor. The parent page for WOObjectEditor is also the return page for WOObjectSelector's buttons. This page, having received the required values from WOObjectSelector, will set the values needed by WOObjectEditor using API bindings. (Note that assignments specified by API bindings are performed behind the scenes by WebObjects using key-value coding.) Finally, the additional information needed by WOObjectEditor, such as the name of the entity we are working with, will be passed through from the parent page of WOObjectSelector using the same techniques.

Figure 3 neatly summarizes the flow of information and the methods for communicating between components. The parent page of WOObjectSelector passes to it the global context, namely entityName and attributeKey, plus the information the selector specifically needs using API bindings. WOObjectSelector then passes the global context along with other specifics that WOObjectEditor needs, such as the user's selection, to the latter's parent using key-value coding. Finally, the editor component's parent passes the information to it using API bindings.

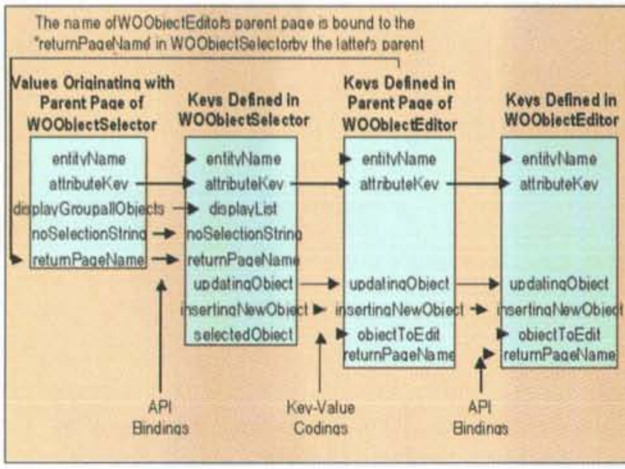


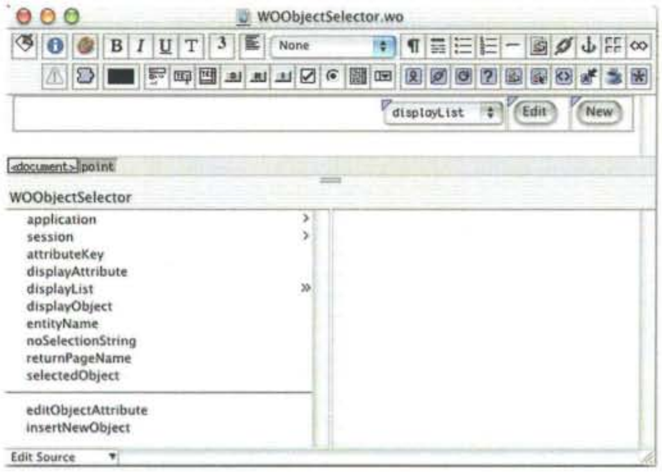
Figure 3.

Now let's consider the details of each component.

WOOBJECTSELECTOR

Figure 4 shows the layout and keys for WOObjectSelector. The component consists of a WOPopUpButton and two submit buttons within a table. The keys attributeKey, displayList, entityName, noSelectionString and returnPageName were mentioned previously. Their values are passed into the component by its parent.

The other keys, displayAttribute, displayObject and selectedObject are used locally. displayAttribute is a method that returns the string for each item displayed in the pop up menu; it uses key-value coding to retrieve the string using attributeKey as the key (Listing 1). The key displayObject is the local EOGenericRecord on which the display list iterates and selectedObject stores the user's selection as an EOGenericRecord. Implicit in this is that a WODisplayGroup's array of EOGenericRecords, has been bound to displayList by the parent. The actions editObjectAttribute and insertNewObject are bound to the "Edit" and "New" buttons, respectively.


WOObjectSelector.wo

None

displayList Edit New

document point

WOObjectSelector

application
session
attributeKey
displayAttribute
displayList
displayObject
entityName
noSelectionString
returnPageName
selectedObject

editObjectAttribute
insertNewObject

Edit Source

Figure 4: Details of WOObjectSelector

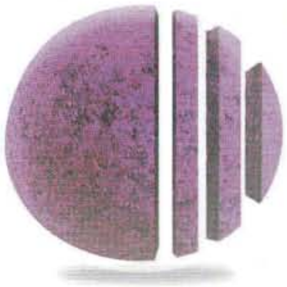
Figure 5 shows the API editor view of WOObjectSelector with the five keys that must be bound by a parent component. Figure 6 shows how WOObjectSelector is embedded in a form element within its parent page, "Main", in the demo application, and the bindings implemented therein. (Be sure to set up the form for multiple submit buttons.) There is only one key declared in Main, namely a WODisplayGroup associated with the example application's reference table, called "Country", whose attribute of interest is "country".

Listings 2 and 3 show the implementation of the actions invoked by WOObjectSelector's buttons. Each simply creates an instance of the return page using the value bound to returnPageName and then invokes takeValueForKey on the page to set the values that WOObjectEditor will eventually need. As mentioned earlier, every child class of WOComponent inherits this key-value coding method.

There is no custom code for the Main class other than the WODisplayGroup variable countryDisplayGroup, which is bound to an entity called Country in the data model for this example application.

YOU'VE GOT YOUR NEW MAC. GREAT. NOW, ALL YOU WANT TO DO IS PUBLISH A SIMPLE, YET ELEGANT WEBSITE, CATALOG AND POST YOUR PICTURES, DO SOME PUBLISHING, DO A LITTLE DRAWING, MAKE A FEW PDF FILES, ARCHIVE SOME DOCUMENTS TO SEND TO YOUR MOTHER, CUT UP SOME BUTTONS FOR THAT ELEGANT WEBSITE, ANIMATE THEM, & TRACK HOW MUCH TIME IT TOOK YOU TO DO IT ALL.

AND YOU WANT TO DO IT FOR LESS THAN $300.

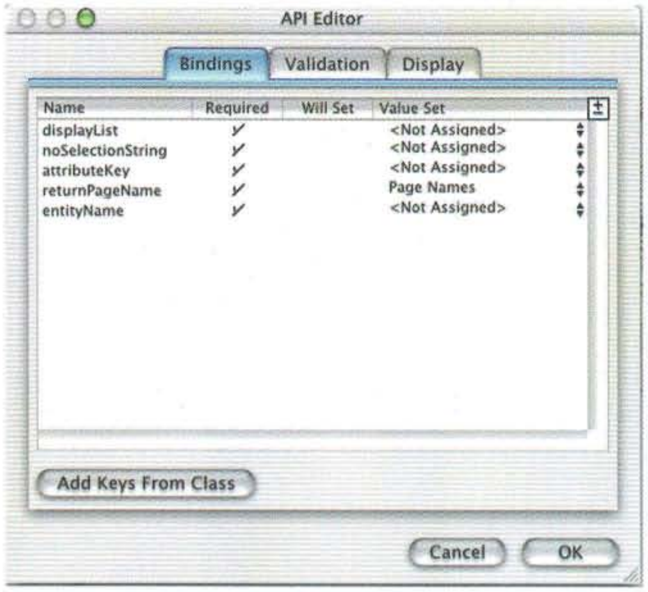


Create™ GifFun™ PackUp&Go™ SliceAndDice™ PhotoToWeb™ TimeEqualsMoney™ PStill™ StampinStone™

STONE STUDIO™ $299

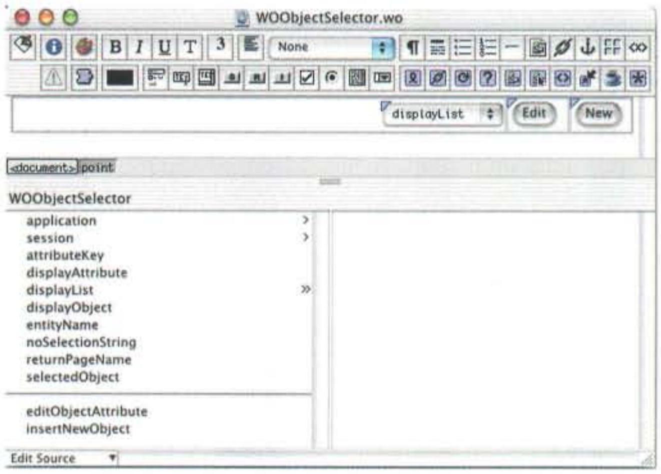
DOWNLOAD NOW OR PURCHASE CD WWW.STONE.COM

STONE STUDIO.™ 8 APPS THAT SET YOU FREE TO DRAW, TWEAK, PROCESS AND PUBLISH YOUR IDEAS. ON PAPER OR ON THE WEB.



Name	Required	Will Set	Value Set
displayList	✓		<Not Assigned>
noSelectionString	✓		<Not Assigned>
attributeKey	✓		<Not Assigned>
returnPageName	✓		Page Names
entityName	✓		<Not Assigned>

Figure 5: API Editor view of WOObjectSelector



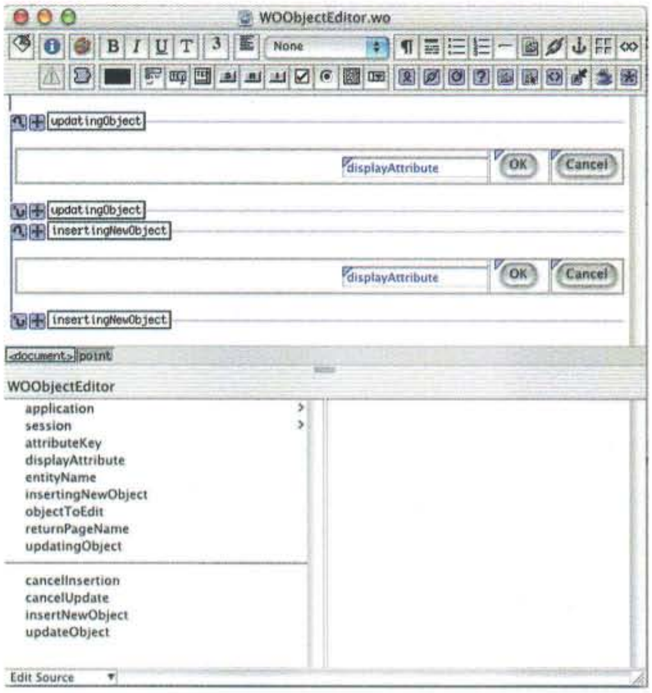
```
WOObjectSelector
application
session
attributeKey
displayAttribute
displayList
displayObject
entityName
noSelectionString
returnPageName
selectedObject

editObjectAttribute
insertNewObject
```

Figure 6: WOObjectSelector laid out in its parent page

WOObjectEditor

Figure 7 shows the layout and keys for WOObjectEditor. The component consists of two WOConditionals, one for the case when the user is updating a record the other for creating a new record. Each conditional consists of a WOTextField and two submit buttons. Four actions are declared in Figure 7, one for each of the buttons. Recall Figure 2. Although the component has four buttons, the user only sees two depending on the choice they made from the selector component.

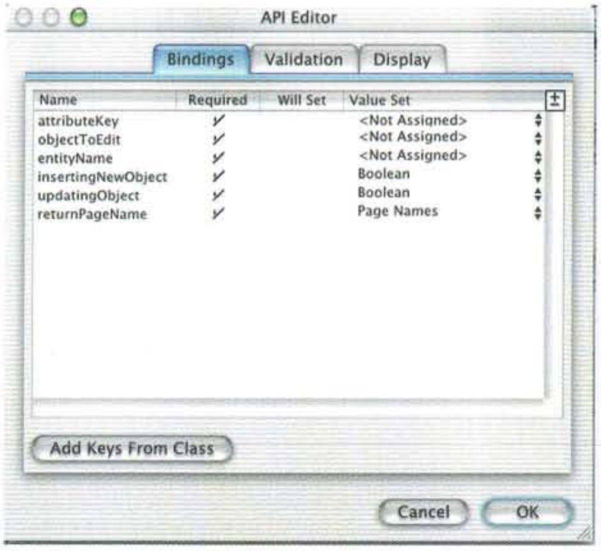
The keys attributeKey, entityName, insertingNewObject, updatingObject and objectToEdit were mentioned earlier. WOObjectSelector provides their values via the editor component's parent using the aforementioned key-value coding and API binding techniques. The editor's parent component provides the value for returnPageName. In our example we simply return to the Main page. As in WOObjectSelector, the key displayAttribute is a method that returns the display string corresponding to the attribute of objectToEdit that we are updating.



```
WOObjectEditor
application
session
attributeKey
displayAttribute
entityName
insertingNewObject
objectToEdit
returnPageName
updatingObject

cancelInsertion
cancelUpdate
insertNewObject
updateObject
```

Figure 7: Details of WOObjectEditor

Figure 8 displays the API Editor view of WOObjectEditor with the six keys that must be bound by a parent component. Figure 9 shows how WOObjectEditor is embedded in its parent page, "CountryEditor", in the demo application and the bindings implemented therein. Notice how the Boolean values insertingNewObject and updatingObject are used not only by WOObjectEditor but also by the enclosing page itself to vary the prompt depending on the action the user selected.



Name	Required	Will Set	Value Set
attributeKey	✓		<Not Assigned>
objectToEdit	✓		<Not Assigned>
entityName	✓		<Not Assigned>
insertingNewObject	✓		Boolean
updatingObject	✓		Boolean
returnPageName	✓		Page Names

Figure 8: API Editor view of WOObjectEditor

"Without a doubt, the Premiere Resource Editor for the Mac OS ... A wealth of time-saving tools."
– MacUser Magazine Eddy Awards

"A distinct improvement over Apple's ResEdit."
– MacTech Magazine

"Every Mac OS developer should own a copy of Resorcerer."
– Leonard Rosenthol, Aladdin Systems

"Without Resorcerer, our localization efforts would look like a Tower of Babel. Don't do product without it!"
– Greg Galanos, CEO and President, Metrowerks

"Resorcerer's data template system is amazing."
– Bill Goodman, author of Smaller Installer and Compact Pro

"Resorcerer Rocks! Buy it, you will NOT regret it."
*– Joe Zobkiw, author of **A Fragment of Your Imagination***

"Resorcerer will pay for itself many times over in saved time and effort."
– MacUser review

"The template that disassembles 'PICT's is awesome!"
– Bill Steinberg, author of Pyro! and PBTools

"Resorcerer proved indispensible in its own creation!"
– Doug McKenna, author of Resorcerer

MACUSER EDITORS' CHOICE AWARD 1993 WINNER

Accelerated for Power Macintosh

Resorcerer® 2

Version 2.0

The Resource Editor for the Mac™ OS Wizard

ORDERING INFO

Requires System 7.0 or greater, 1.5MB RAM, CD-ROM

Standard price: $256 (decimal)
Website price: $128 - $256
(Educational, quantity, or other discounts available)

Includes: Electronic documentation
60-day Money-Back Guarantee
Domestic standard shipping

Payment: Check, PO's, or Visa/MC
Taxes: Colorado customers only

Extras (call, fax, or email us):
COD, FedEx, UPS Blue/Red, International Shipping

MATHEMÆSTHETICS, INC.
PO Box 298
Boulder, CO 80306-0298 USA
Phone: (303) 440-0707
Fax: (303) 440-0504
resorcerer@mathemaesthetics.com

New in 2.0:

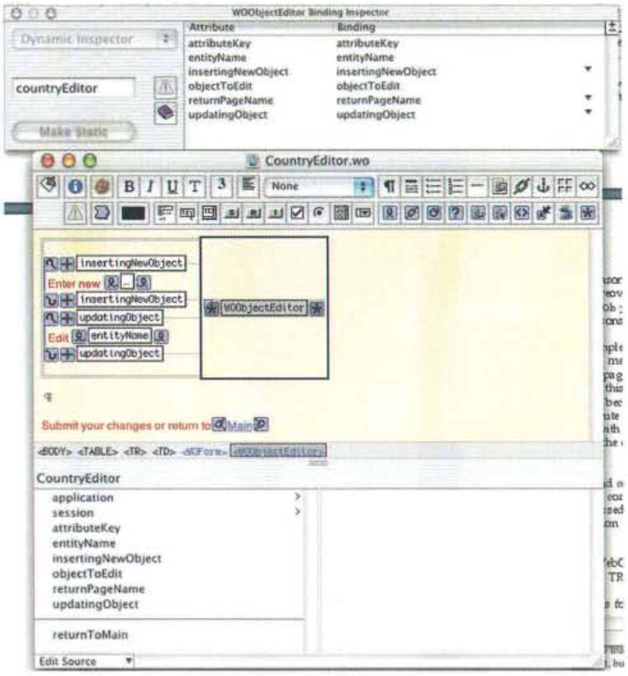
- **Very fast, HFS browser for viewing file tree of all volumes**
- **Extensibility for new Resorcerer Apprentices (CFM plug-ins)**
- **New AppleScript Dictionary ('aete') Apprentice Editor**
- **MacOS 8 Appearance Manager-savvy Control Editor**
- **PowerPlant text traits and menu command support**
- **Complete AIFF sound file disassembly template**
- **Big-, little-, and even mixed-endian template parsing**
- **Auto-backup during file saves; folder attribute editing**
- **Ships with PowerPC native, fat, and 68K versions**

- Fully supported; it's easier, faster, and more productive than ResEdit
- Safer memory-based, not disk-file-based, design and operation
- All file information and common commands in one easy-to-use window
- Compares resource files, and even **edits your data forks** as well
- Visible, accumulating, editable scrap
- Searches and opens/marks/selects resources by text content
- Makes global resource ID or type changes easily and safely
- Builds resource files from simple Rez-like scripts
- Most editors DeRez directly to the clipboard
- All graphic editors support screen-copying or partial screen-copying
- Hot-linking Value Converter for editing 32 bits in a dozen formats
- Its own 32-bit List Mgr can open and edit very large data structures
- Templates can pre- and post-process any arbitrary data structure
- Includes nearly 200 templates for common system resources
- TMPLs for Installer, MacApp, QT, Balloons, AppleEvent, GX, etc.
- Full integrated support for editing color dialogs and menus
- Try out balloons, 'ictb's, lists and popups, even create C source code
- Integrated single-window Hex/Code Editor, with patching, searching
- Editors for cursors, versions, pictures, bundles, and lots more
- Relied on by thousands of Macintosh developers around the world

To order by credit card, or to get the latest news, bug fixes, updates, and apprentices, visit our website...

www.mathemaesthetics.com

Listings 4 and 5 show the implementations for the actions insertNewObject and updateObject. The former creates a new instance of the entity specified by entityName then sets the value of the specified attribute using key-value coding. Upon inserting the new record into the default editing context, it saves the changes. The updateObject method simply sets the new value of objectToEdit using key-value coding, then saves the changes.



Attribute	Binding
attributeKey	attributeKey
entityName	entityName
insertingNewObject	insertingNewObject
objectToEdit	objectToEdit
returnPageName	returnPageName
updatingObject	updatingObject

Figure 9: WOObjectEditor laid out in a its parent page, CountryEditor

Enhancements and Improvements

Our pair of components is already very useful and highly reusable. However, there are possibilities for improvements. One concerns validation and associated exception handling as follows:

The two action methods insertNewObject and updateObject have built-in validation rules; as written they each enforce a non-null value for every attribute of every entity. This may not be unreasonable in the case of reference tables, but it isn't as flexible as it could be. Moreover, there is no exception handling for the possibility that either insertObject or saveChanges fails. These operations could fail for a variety of reasons including constraints built in to the data model(s) used in a real application.

There are two obvious alternatives to implementing improved validation and exception handling. One is to override the method validationFailedWithException in the page that encloses WOObjectEditor. Every component inherits this method. One would probably implement it in the editor component's parent because the parent knows the context in which validation occurs. Another route would be to subclass and provide the subclass with the required context. I probably would choose the former path since the context is already known there.

Summary

This article has illustrated the design and construction of reusable WebObjects components and inter-component communication using key-value coding. The fully functional components discussed herein are widely applicable to the maintenance of reference tables, a common situation.

Bibliography and References

The interested reader may find the following books useful. Of the three, Ruzek's book is the best, in my opinion. Unfortunately, it is based on WebObjects version 4.5 and emphasizes development under Windows. Nevertheless, much is applicable and the examples are pretty good. DeMan et al's book is based on version 5 and incorporates references to OSX. Their discussion of validation, key-value coding and other important topics is very good. However, the book is a bit rough around the edges in some places perhaps reflecting its multiple authorship and the need for a bit more editing. Feiler's book is the least helpful. After a very long (60 pages) and general introduction, the book finally gets around to discussing OpenBase. For many topics the author merely recapitulates Apple's documentation, for others he provides only the most cursory treatment and no concrete examples.

- DeMan, Michael, Frederico, Gustavo, et. al. *Professional WebObjects 5.0 with Java*, Wrox Press Ltd., Birmingham, UK, 2001.
- Ruzek, George. *WebObjects Web Application Development Kit*, Sams Publishing, Indianapolis, 2001
- Feiler, Jesse. *Building WebObjects 5 Applications*, McGraw-Hill/Osborne, Berkeley, 2002.

Listing 1: WOObjectSelector.java

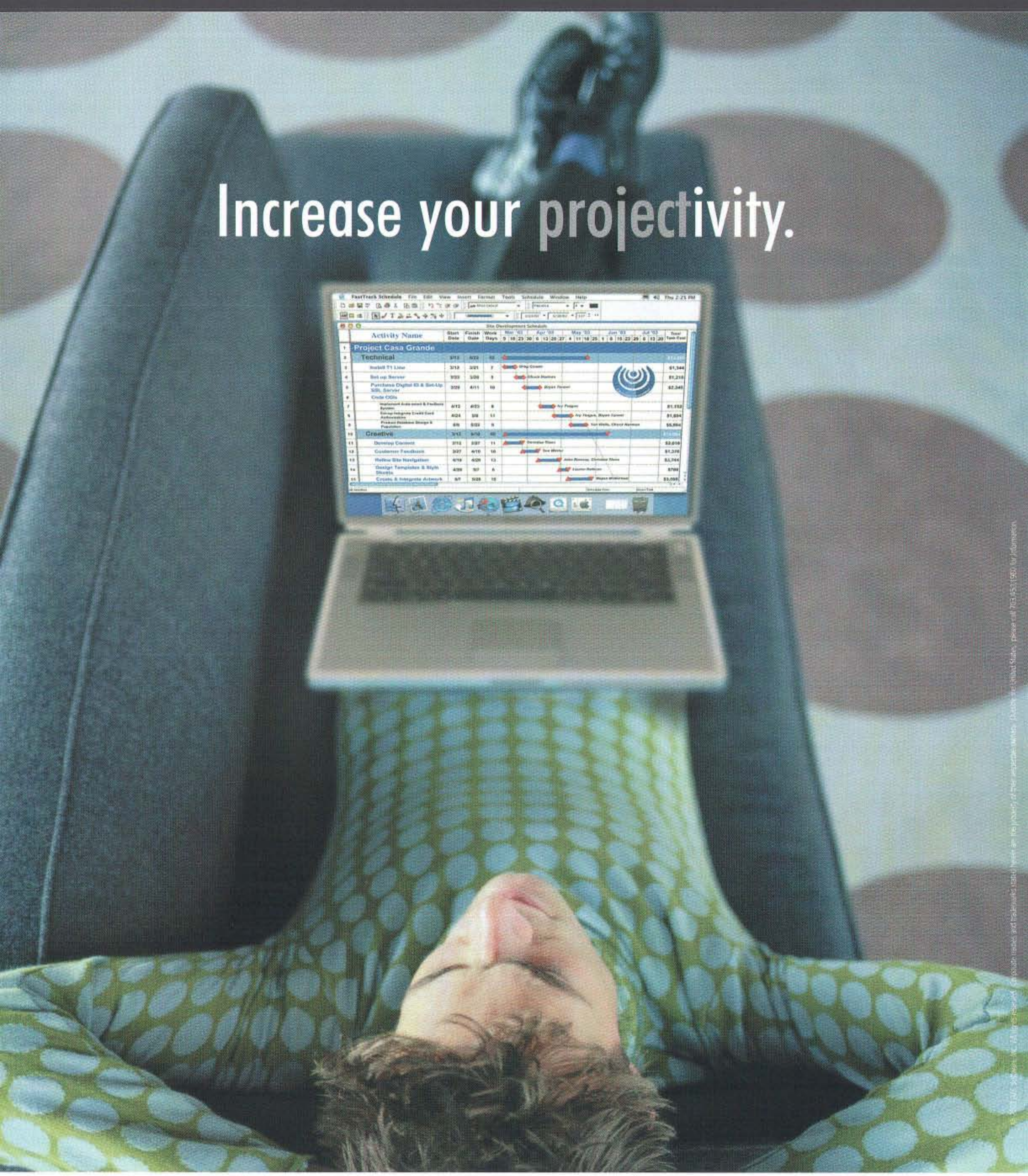
displayAttribute

```
public String displayAttribute(){
  return (String) displayObject.valueForKey(attributeKey);
}
```

Listing 2: WOObjectSelector.java

editObjectAttribute

```
public WOComponent editObjectAttribute(){
  /*
    The user needs to select some value to edit;
    if not, do nothing.
  */
  if (selectedObject == null) return null;
  WOComponent nextPage =
          (WOComponent)pageWithName(returnPageName);
  nextPage.takeValueForKey(entityName,"entityName");
  nextPage.takeValueForKey(selectedObject,"objectToEdit");
  nextPage.takeValueForKey(attributeKey,"attributeKey");
  nextPage.takeValueForKey(Boolean.TRUE,"updatingObject");\
nextPage.takeValueForKey(Boolean.FALSE,
                                    "insertingNewObject");
  return nextPage;
```



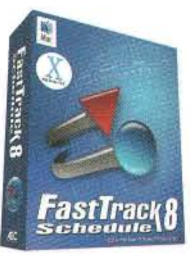
FastTrack Schedule File Edit View Insert Format Tools Schedule Window Help — Thu 2:25 PM

Site Development Schedule

	Activity Name	Start Date	Finish Date	Work Days	Mar '03 – Jul '03	Total Task Cost
1	Project Casa Grande					
2	Technical	3/12	[illegible]	[illegible]		[illegible]
3	Install T1 Line	3/12	3/21	7	[illegible]	$1,344
4	Set up Server	3/22	3/28	5	[illegible]	$1,218
5	Purchase Digital ID & Set-Up SSL Server	3/29	4/11	10	[illegible]	$2,340
6	Code CGIs					
7	Implement Auto-email & Feedback System	4/12	4/23	8	Ivy Teague	$1,152
8	Set-up Integrate Credit Card Authorization	4/24	5/8	11	Ivy Teague, [illegible]	$1,884
9	Product Database Design & Population	5/9	5/22	9	[illegible]	$5,894
10	Creative	3/12	[illegible]	[illegible]		[illegible]
11	Develop Content	3/12	3/27	11	[illegible]	$2,616
12	Customer Feedback	3/27	4/10	10	[illegible]	$1,376
13	Refine Site Navigation	4/10	4/26	13	[illegible]	$3,744
14	Design Templates & Style Sheets	4/29	5/7	6	[illegible]	$798
15	Create & Integrate Artwork	5/7	5/28	15	[illegible]	$3,098

Be productive. Stay productive.

Introducing FastTrack Schedule 8. Redesigned for Mac OS X and packed with new productivity features and a bold Aqua interface–FastTrack Schedule 8 has all the tools you need to ensure project success. For a free demo version or to order, call us today at 800.450.1983.



Built for Mac OS X

www.fasttrackschedule8.com

AEC SOFTWARE

Fetch

Mac's best friend.

Fetchsoftworks.com

Version 4.0.2 now available.

```
}
```

Listing 3: WOObjectSelector.java

insertNewObject

```
public WOComponent insertNewObject(){
  WOComponent nextPage =
          (WOComponent)pageWithName(returnPageName);
  nextPage.takeValueForKey(entityName,"entityName");
  nextPage.takeValueForKey(null,"objectToEdit");
  nextPage.takeValueForKey(attributeKey,"attributeKey");
  nextPage.takeValueForKey(Boolean.FALSE,"updatingObject");
  nextPage.takeValueForKey(Boolean.TRUE,
                                "insertingNewObject");
  return nextPage;
}
```

Listing 4: WOObjectEditor.java

insertNewObject

```
public WOComponent insertNewObject() {
  if (displayAttribute!=null &&
            displayAttribute.length()>0){
    /*
      the user has entered a non-blank string;
      create a new object
    */
    EOClassDescription description =
      EOClassDescription.classDescriptionForEntityName(
                                          entityName);
    EOEnterpriseObject newObject =
      description.createInstanceWithEditingContext(null,
                                          null);
    newObject.takeValueForKey(displayAttribute,
                                        attributeKey);
    EOEditingConext dec =
                this.session().defaultEditingContext();
    dec.insertObject(newObject);
    dec.saveChanges();
  }
  WOComponent nextPage =
            (WOComponent)pageWithName(returnPageName);
  return nextPage;
}
```

Listing 5: WOObjectEditor.java

updateObject

```
public WOComponent updateObject(){
  if (displayAttribute!=null &&
            displayAttribute.length()>0){
    // the user has entered a non-blank string
    objectToEdit.takeValueForKey(displayAttribute,
                                        attributeKey);
    this.session().defaultEditingContext().saveChanges();
  }
  WOComponent nextPage =
            (WOComponent)pageWithName(returnPageName);
  return nextPage;
}
```

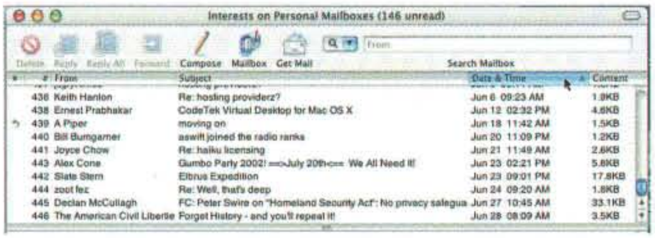
By - Andrew C. Stone

Takes All Sorts

Make Your TableViews Autosort!

One of the coolest pieces of object technology in the Cocoa bag of tricks is the NSTableView, and its daughter, NSOutlineView. The tableView lets you display a list of items with various properties. As in a spreadsheet, it displays rows of information - anything from simple text to graphics or even quicktime movies! Once you learn the fundamentals of the tableView and the outlineView, they'll quickly become some of your favorite user interface objects.

This article will take the NSTableView further, and show you how to add "auto sort" to your tableView, and the objects it represents. It assumes you have read the documentation for NSTableView and its companion class NSTableColumn and perhaps already have an app with an NSTableView of your own. You can try out this sort idea in Apple's Mail. A Mail Box shows a tableView of the list of messages, sender, subject, date, size, etc:



#	From	Subject	Date & Time	Content
436	Keith Hanlon	Re: hosting providerz?	Jun 6 09:23 AM	1.8KB
438	Ernest Prabhakar	CodeTek Virtual Desktop for Mac OS X	Jun 12 02:32 PM	4.6KB
439	A Piper	moving on	Jun 18 11:42 AM	1.5KB
440	Bill Bumgarner	aswift joined the radio ranks	Jun 20 11:09 PM	1.2KB
441	Joyce Chow	Re: haiku licensing	Jun 21 11:49 AM	2.6KB
443	Alex Cone	Gumbo Party 2002! ==>July 20th<== We All Need It!	Jun 23 02:21 PM	5.8KB
442	Slate Stern	Elbrus Expedition	Jun 23 09:01 PM	17.8KB
444	zoot fez	Re: Well, that's deep	Jun 24 09:20 AM	1.8KB
445	Declan McCullagh	FC: Peter Swire on "Homeland Security Act": No privacy safegua	Jun 27 10:45 AM	33.1KB
446	The American Civil Libertie	Forget History - and you'll repeat it!	Jun 28 08:09 AM	3.5KB

***Figure 1.** You can play with quick sort feature in Apple's Mail application.*

When you click on the title cell of a column, an arrow appears to show direction of the sort, the cell is highlighted, and the messages are sorted on that key. If you click the title cell of that same column again, it toggles the sort to reverse its direction: if it was an ascending sort, now it's a descending sort. Sorting mail by subject is particularly interesting for deleting spam en masse!

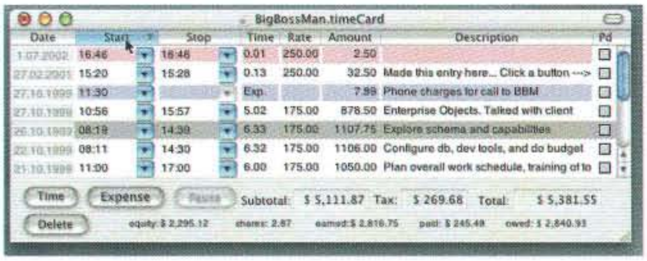
When Time Equals Money...

When preparing TimeEqualsMoney 2.0 <http://www.stone.com/TimeEqualsMoney/> this spring, I decided users would love to be able to sort their time and expense entries on any key, in either direction with a simple click. So I reread the documentation Apple provides on the NSTableView class. I highly recommend actually reading the .h files! It's amazing how much stuff is already anticipated for you! There is a delegate method, - (void) tableView:(NSTableView*)tv didClickTableColumn:(NSTableColumn *)tableColumn. So, it seemed, I simply had to implement this method in my NSWindowController subclass, which is architecturally a sound choice for the NSTableView delegate. However, the method was simply not being called when I clicked on the column header cell. It turns out, and I noted this heavily in my code, that if neither column reodering nor column selection is selected in Interface Builder's NSTableView inspector, then the method is not called. Turning on column reordering was a simple fix and a desired feature! You can, of course, also call one of these methods programmatically with a parameter of YES:

```
- (void)setAllowsColumnReordering:(BOOL)flag;
- (void)setAllowsColumnSelection:(BOOL)flag;
```

Following on the Model-View-Controller pattern that I have discussed in previous articles on Cocoa architecture, I decided to store the sort column, and its direction in my NSDocument subclass, as well as make it responsible for the sorting of its items. This way, you can undo the sort, automatically mark the document as unsaved as well as save the sort between sessions. The user clicks the tableView column header, the window controller then sets the document's sort key, or direction if the key was already set to that column. Then the document tells itself to sort, which tells its NSMutableArray of entries to sort its contents. Finally, the document tells its window controllers to reload the tableView, which updates the interface to display the new sort order:

Andrew Stone, andrew@stone.com, is Chief Chef at Stone Design, http://www.stone.com, has this to say about Cocoa and Stone Studio with all due respect to Gary Snyder's Turtle Island, "It's objects all the way down" .



BigBossMan.timeCard

Date	Start	Stop	Time	Rate	Amount	Description	Pd
1.07.2002	16:46	16:46	0.01	250.00	2.50		☐
27.02.2001	15:20	15:28	0.13	250.00	32.50	Made this entry here... Click a button --->	☐
27.10.1999	11:30		Exp.		7.99	Phone charges for call to BBM	☐
27.10.1999	10:56	15:57	5.02	175.00	878.50	Enterprise Objects. Talked with client	☐
26.10.1999	08:19	14:39	6.33	175.00	1107.75	Explore schema and capabilities	☐
22.10.1999	08:11	14:30	6.32	175.00	1106.00	Configure db, dev tools, and do budget	☐
21.10.1999	11:00	17:00	6.00	175.00	1050.00	Plan overall work schedule, training of to	☐

Subtotal: $ 5,111.87 Tax: $ 269.68 Total: $ 5,381.55

equity: $ 2,295.12 shares: 2.87 earned:$ 2,816.75 paid: $ 245.48 owed: $ 2,840.93

Figure 2. This tableView lets you click a column header to sort on that key - up or down.

Model Behavior

The implementation of the actual sort is trivial: just ask the list to sort itself based on asking its elements to sort using a selector: [list sortUsingSelector:@selector(compare:)];

During the list's sorting procedure, it repeatedly calls, in this case, compare: on one list item against another.

I like to just reuse the Kit's compare: which is implemented in NSString, NSDate, NSCell, NSValue, etc:

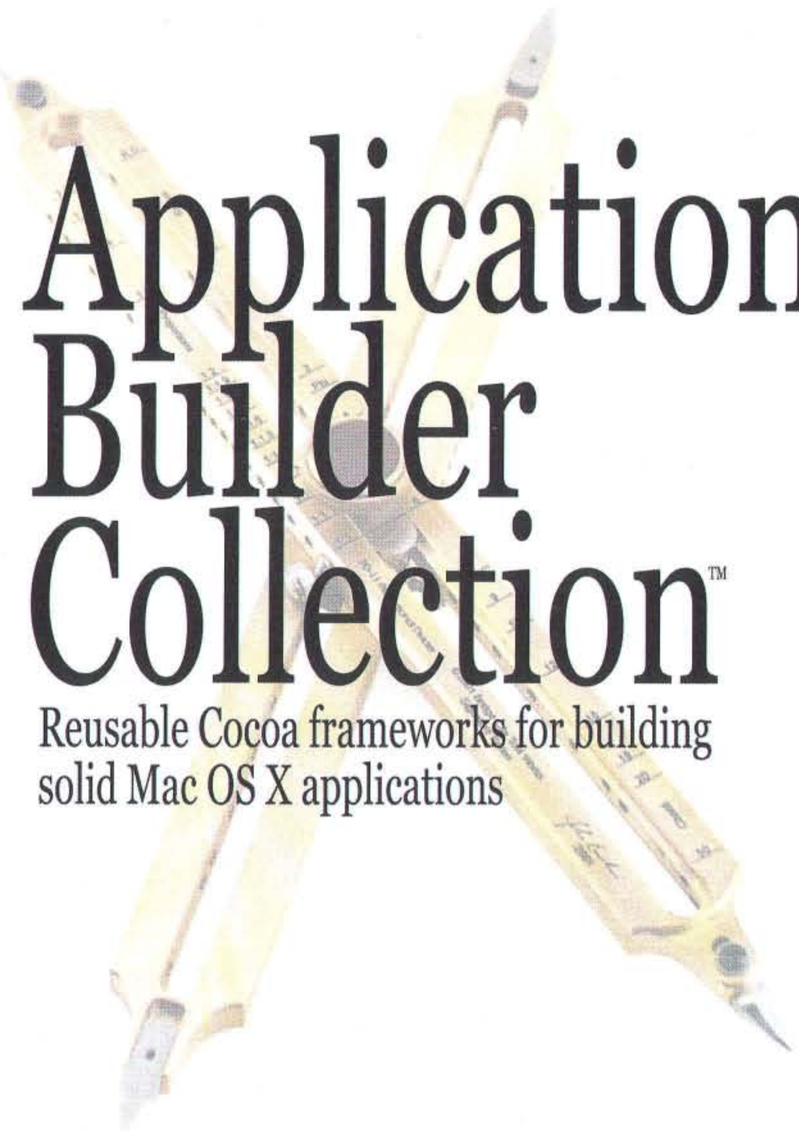
```
- (NSComparisonResult)compare:(id)other;
```

The NSComparisonResult has these three values: NSOrderedAscending = -1, NSOrderedSame = 0, NSOrderedDescending = 1. Since so many objects reply to compare:, you might implement it something like this in your data object, given that the sort key is the same string as the column identifiers, and you have various numbers, dates, strings and even booleans:

```
// the Data MODEL - the document has a list of these entities:

@interface PieceWork : NSObject <NSCopying>
{
    NSCalendarDate *startTime;
    NSString *description;
    NSTimeInterval timeOnJob;
    float _rate;
    BOOL paid;
    id owner;
...
}
- (NSComparisonResult)compare:(id)other;
- (void)setOwner:(id)ownsMe;
- (NSCalendarDate *)startTime;
- (NSTimeInterval)timeOnJob;
- (NSString *)workDescription;
- (float)hoursOnJob;
- (float)rate;
- (BOOL)paid;
...
@end

@implementation PieceWork
...

- (NSComparisonResult)compare:(id)other {
    BOOL isDescending = [owner sortIsDescending];
    NSString *key = [owner sortColumn];

  // now determine which one is first based on sort direction:
    PieceWork *first = isDescending ? other : self;
    if (first == other) other = self;

    if ([key isEqualToString:@"TimeOnJob"]) {
        return [[NSNumber numberWithFloat:[other
timeOnJob]] compare: [NSNumber
numberWithFloat:[first timeOnJob]]];
    } else if ([key isEqualToString:@"Date"]) {
        return [[other startTime] compare: [first
startTime]];
    } else if ([key
isEqualToString:@"Description"]) {
// Very handy macro:
#define IS_NULL(s) (!s || [s isEqualToString:@""])
            if (IS_NULL([other workDescription]))
return NSOrderedDescending;
            if (IS_NULL([first workDescription]))
return NSOrderedAscending;
// people prefer upper and lower case mixed, unlike UNIX:
        return [[other workDescription]
caseInsensitiveCompare:[first workDescription]];
    } else if ([key isEqualToString:@"Paid"]) {
        return [[NSNumber numberWithBool:[other
paid]] compare:[NSNumber numberWithBool:[first
paid]]];
    }
    return 0;
}
@end
```

Here are the relevant instant variables and methods in the NSDocument subclass:

```
@interface MyDocument : NSDocument {
    NSMutableArray *_workList;
    NSString *_sortColumn;
    BOOL _sortIsDescending;
.   ...
}
- (void)sort;
- (NSArray *)workList;
- (void)setSortColumn:(NSString *)identifier;
- (NSString *)sortColumn;
- (void)setSortIsDescending:(BOOL)whichWay;
- (BOOL)sortIsDescending;
...
@end

@implementation MyDocument
...

- (void)sort:(NSMutableArray *)list {
    [list
makeObjectsPerformSelector:@selector(setOwner:)
withObject:self];
    [list sortUsingSelector:@selector(compare:)];
// asks us for how!!
}

- (void) sort {
    if (NOT_NULL(_sortColumn)) {
        [self sort:_workList];
        [[self windowControllers]
makeObjectsPerformSelector:@selector(updateTotalsAnd
Reload)];
    }
}

- (void)setSortColumn:(NSString *)identifier {
    if (![identifier isEqualToString:_sortColumn])
{
        [[[self undoManager]
prepareWithInvocationTarget:self]
setSortColumn:_sortColumn];
    [_sortColumn release];
    _sortColumn = [identifier copyWithZone:[self
zone]];
        [[self undoManager]
setActionName:NSLocalizedStringFromTable(@"Sort",@"T
imeCard",@"title of undo the sort action")];
    }
}

- (NSString *)sortColumn {
    return _sortColumn;
}
```



Application Builder Collection™

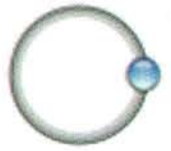
Reusable Cocoa frameworks for building solid Mac OS X applications

- Port to Cocoa faster
- Save thousands by reducing coding and testing time
- Spend your time developing the features that make your software stand out
- Quickly create professional looking applications
- Utilize object-oriented technology for easy customization and reuse
- No deployment licenses or royalties!

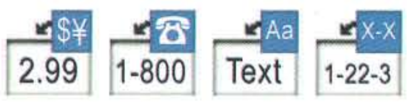
Collection Includes:

- Rotational Slider
- Formatters for Currency, Phone, Text, and Custom Fields
- Draggable Cocoa Cells for Tables
- Calendar View and Popup
- URL Clickable Field
- Twiddle View
- Archive Builder Application
- And More . . .

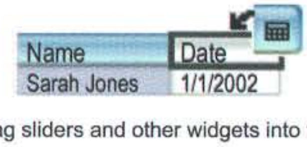
Learn more at www.jiiva.com



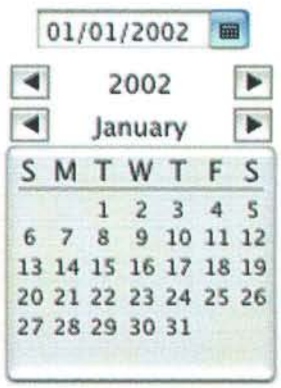
rotate images and more



control data input and presentation

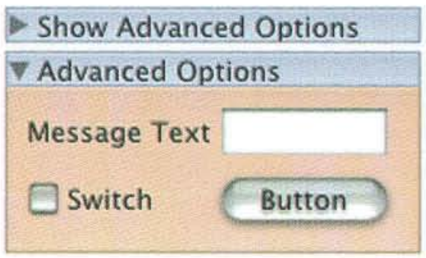


drag sliders and other widgets into tables



add calendar popups and views

turn text fields into clickable URLs



create complex interfaces with versatile twiddles

Mac

Built for Mac OS X

Jiiva, Inc.
www.jiiva.com/abc

```
- (void)setSortIsDescending:(BOOL)whichWay {
    if (whichWay != _sortIsDescending) {
        [[[self undoManager]
prepareWithInvocationTarget:self]
setSortIsDescending:_sortIsDescending];
    _sortIsDescending = whichWay;
        [[self undoManager]
setActionName:NSLocalizedStringFromTable(@"Sort
Direction",@"TimeCard",@"title of undo the sort up
or down action")];
    }
}

- (BOOL)sortIsDescending {
    return _sortIsDescending;
}

- (NSArray *)workList { return _workList; }

- (void)setWorkList:(NSMutableArray *)list {
    [[[self undoManager]
prepareWithInvocationTarget:self]
setWorkList:_workList];
    [_workList release];
    if (NOT_NULL(_sortColumn)) [self sort:list];
    _workList = [list mutableCopyWithZone:[self
zone]];
    [_workList
makeObjectsPerformSelector:@selector(setOwner:)
withObject:self];
    [[self windowControllers]
makeObjectsPerformSelector:@selector(updateTotalsAnd
Reload)];
    [[self undoManager]
setActionName:NSLocalizedStringFromTable(@"Work
Change", @"TimeCard", @"Action name for changing
worklist")];
}

// And don't forget to actually archive and read the sort info
// I highly recommended using human readable XML dictionaries:

- (NSMutableDictionary *)workDocumentDictionary {
    NSMutableDictionary *doc = [NSMutableDictionary
dictionary];
        unsigned i, c = [_workList count];
        if (NOT_NULL(_sortColumn)) [doc
setObject:_sortColumn forKey:SortNameKey];
        if (_sortIsDescending) [doc
setObject:@"YES" forKey:SortDescendingKey];
        ...
       return doc;
}

- (BOOL)loadDataRepresentation:(NSData *)data
ofType:
                                   (NSString *)type {
    if ([type isEqualToString:DocumentType]) {
 NSDictionary *doc = [self
workDocumentDictionaryFromData:data];
        id obj;
        ...
        obj = [doc objectForKey:SortNameKey];
        if (obj) {
            [self setSortColumn:obj];
        }
      obj = [doc objectForKey:SortDescendingKey];
        if (obj) {
            _sortIsDescending = [@"YES"
isEqualToString:obj];
        }
        return YES;
  }
  return NO;
}

@end
```

Control Freak

Finally, we come to the meat of the matter in our NSWindowController subclass which is our NSTableView's dataSource as well as delegate.

```
@interface JobWindowController : NSWindowController
{
    IBOutlet NSTableView *tableView;
    NSImage *descendingSortingImage;
    NSImage *ascendingSortingImage;
    ...
}
```

We have work to do in our window initialization method that gets called when the Interface has loaded, awakeFromNib. We want to get the images that we shall display when a sort is made. Using class-dump as described in my iPhoto Exporter bundle article <http://www.omnigroup.com/~nygard/Projects/index.html>, you might note that there are hidden, private Apple internal API to access the images to indicate ascending or descending sorts! Since one should absolutely never rely on hidden API to not dissolve into the ether in subsequent system releases, always bracket the use of these methods with a "respondsToSelector:" query, and provide backup images of your own:

```
- (void)awakeFromNib
{
  // private images:
    if ([[NSTableView class]
respondsToSelector:@selector(_defaultTableHeaderSort
Image)])
    ascendingSortingImage = [[NSTableView class]
_defaultTableHeaderSortImage];
    else ascendingSortingImage = [[NSImage
imageNamed:@"ascendingSort"] retain];

    if ([[NSTableView class]
respondsToSelector:@selector(_defaultTableHeaderReve
rseSortImage)])
    descendingSortingImage = [[NSTableView class]
_defaultTableHeaderReverseSortImage];
    else descendingSortingImage = [[NSImage
imageNamed:@"descendingSort"] retain];
...
}
```

If you allow users to move and resize columns, then you definitely want to additionaly call these two methods in awakeFromNib - this will reestablish their preferences and give the system a name to save the column order and sizes.

```
    [tableView setAutosaveTableColumns:YES];
    [tableView setAutosaveName:@"WorkTable"];
```

So, now we're ready to implement the delegate method which gets called when the column title cell is clicked. The one fine point is that we want to make a note of which item was selected before the sort, so we can reselect it after the sort (its row number may have changed after the sort). We ask the tableView if it already has an image in that column, thus indicating that we need to toggle the direction, otherwise, we'll set the new sort key. The actual work of setting up the tableView's header hilighting is done in

updateTableHeaderToMatchCurrentSort. This is factored into a separate method so we can also call it when loading a document to restore last saved sort state.

```
// BIG COCOA NOTE: if column reordering or column selection is not
on
// then this doesn't get called!!

- (void) tableView:(NSTableView*)tv
didClickTableColumn:(NSTableColumn *)tableColumn {
     NSImage *sortOrderImage = [tv
indicatorImageInTableColumn:tableColumn];
     NSString *columnKey = [tableColumn identifier];
     MyDocument *doc = [self document];
     PieceWork *work = [self selectedWork];
  // If the user clicked the column which already has the sort indicator
  // then just flip the sort order.

     if (sortOrderImage || columnKey == [doc
sortColumn]) {
          [doc setSortIsDescending:![doc
sortIsDescending]];
     } else {
          [doc setSortColumn:columnKey];
     }
     [self updateTableHeaderToMatchCurrentSort];
  // now do it - doc calls us back when done
     [doc sort];
  // but reselect the one previously selected:
     if (work) [self selectWork:work];
}
```

This method updates the state of the NSTableColumn by first clearing all cells, and then setting the correct image and hilighting on the column to sort by, if any:

```
- (void)updateTableHeaderToMatchCurrentSort {
     BOOL isDescending = [[self document]
sortIsDescending];
     NSString *key = [[self document] sortColumn];
     NSArray *a = [tableView tableColumns];
     NSTableColumn *column = [tableView
tableColumnWithIdentifier:key];
     unsigned i = [a count];

     while (i-- > 0) [tableView setIndicatorImage:nil
inTableColumn:[a objectAtIndex:i]];

     if (NOT_NULL(key)) {
          [tableView setIndicatorImage:(isDescending
? ascendingSortingImage:descendingSortingImage)
inTableColumn:column];

          [tableView
setHighlightedTableColumn:column];
     } else [tableView
setHighlightedTableColumn:nil];
}
```

And our helper functions to make the code neat and compact:

```
- (NSArray *)workList {
    return [[self document] workList];
}

- (void)updateTotalsAndReload {
// other ui:
//   [self updateTotals];
     [tableView reloadData];
}

- (PieceWork *)selectedWork {
    NSArray *workList = [self workList];
    int row = [tableView selectedRow];
    if (row > -1 && row < [workList count]) return
[workList objectAtIndex:row];
    return nil;
}

- (void)selectWork:(PieceWork *)work {
    NSArray *a = [self workList];
    unsigned i = [a indexOfObject:work];
    if (i != NSNotFound) [self selectRow:i];
}
```

Devil in the Details

So, now we got sorting working, what sort of problems might we run into? Let's say a user edits the field of an entry in the sorted column. Upon finishing editing, they may have changed the relative order of that entry, requiring a new sort, and a reloading of the tableView again! We can check if we have the sorted column when the data gets set in -tableView:setObjectValue:forTableColumn:row:

```
- (void)tableView:(NSTableView *)tv
setObjectValue:(id)object
forTableColumn:(NSTableColumn *)tc row:(int)row;
{
        NSArray *workList = [self workList];
        NSString *ident = [tc identifier];
        if (tv == tableView) {
             PieceWork *work = [workList
objectAtIndex:row];
             if ([ident isEqual:@"Description"]) {
                  [work setWorkDescription:object];
                  [self sortIfSortedOn:ident
work:work];
             } else if ....
}
```

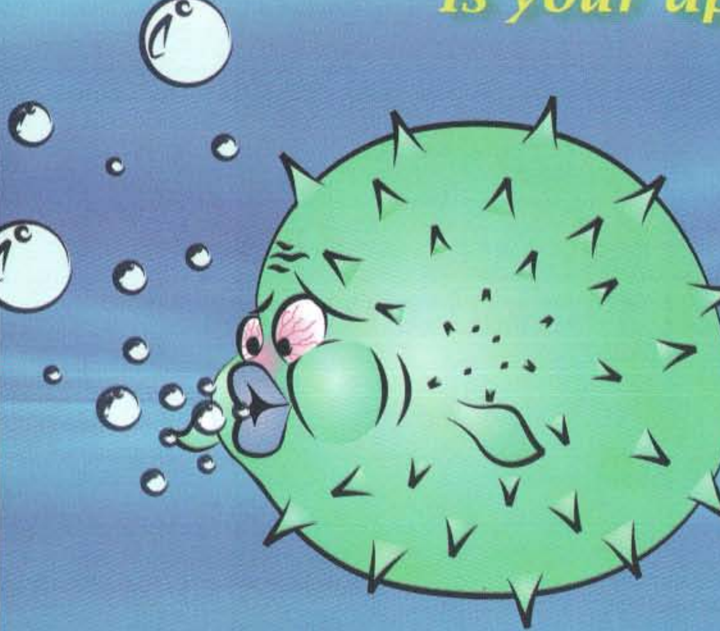
Isn't the devil in the details? Because the tableView will want to "select next" after setObjectValue: is called, the wrong row might be selected if the sort changed the currently selected item's row. By calling this method, we'll first end editing on the tableView so any inadvertent select next is foiled. Note the delay of 0.0 seconds which means to schedule the call of the method after the current event loop finishes:

```
- (void)afterSort:(PieceWork *)work {
    [[self window] makeFirstResponder:[self
window]];
    [self selectWork:work];
}

- (void)sortIfSortedOn:(NSString *)ident
work:(PieceWork *)work {
    if ([[[self document] sortColumn]
isEqualToString:ident])  {
         int oldPosition
         [[self window] makeFirstResponder:[self
window]];
         [[self document] sort];
         [self performSelector:@selector(afterSort:)
withObject:work afterDelay:0.0];
    } else [self updateTotalsAndReload];
}
```

Conclusion

Sorting is something users expect tableViews to do now - and with a little effort, all of your tableViews can sort themselves!

Is your application's heap expanding like a blowfish with a water allergy?*



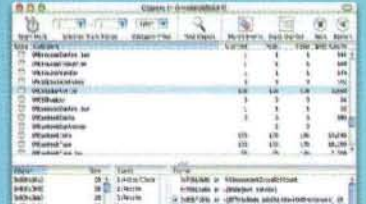
Are users complaining of constant swapping?

Do RAM manufacturers send you birthday cards?

Do you call Jon Rubenstein to ask about terabyte iMacs?

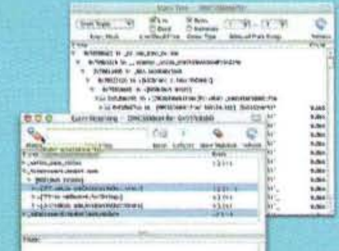
It's not something to discuss in polite company, but you, my friend, just may have a memory leak.

OMNIOBJECTMETER 2.0



OmniObjectMeter has not been tested on blowfish allergies, but it is the best tool to help you seek and destroy memory leaks and expensive allocation operations in your Cocoa programs.

- Track allocations, references, and deallocations of Objective-C, CoreFoundation, and BSD objects in a running process.
- Use our unique matching window to quickly match and hide paired reference count changing events. All that will be left are leaks, zombies, and retain cycles.
- Track down and eliminate transient objects, excessive autoreleases, and other inefficient operations.



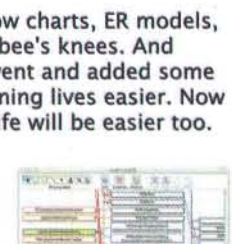
— BUT WAIT, THERE'S MORE —

OMNIGRAFFLE 2.0

Our symbolic drawing application easily generates flow charts, ER models, network diagrams, and much more. We think it's the bee's knees. And being the overachieving developers that we are, we went and added some cool features to OmniGraffle that make our programming lives easier. Now with 2.0 we're sharing them with the world, so your life will be easier too. Aren't we just peachy that way?

Project Builder / Framework Import : drag in a Cocoa project or framework to automatically create a class hierarchy diagram with clickable links to header files.

AppleScript Everywhere : use OmniGraffle as an interface for tasks like bug tracking and scheduling. Attach scripts to graphics to make interactive documents; import and layout data from scriptable applications like FileMaker.



OMNIOUTLINER 2.0

OmniOutliner is a general-purpose outlining and list-making program; we've found it has myriad uses in software development: bug lists, project schedules, pseudocode outlines, etc. Plus a few features just for geeks like us:

"Sample" Import : presents output from Apple's command-line "sample" program (a useful performance monitoring tool) in an easy-to-read hierarchical view. (Just remove the ".txt" extension from the output file and open it in OmniOutliner.)

More AppleScript : 2.0 adds complete AppleScript support, so, for example, you can dump your project schedule to OmniGraffle.

If you've bought a Power Mac or PowerBook since January 2002, you're already sitting in the back of the bus with the cool kids — OmniGraffle 1.1 and OmniOutliner 1.2 are already installed in your Applications folder. But drop by our website and upgrade to the 2.0 versions we talk about here: they're much cooler.

WWW.OMNIGROUP.COM/MACTECH

THE OMNI GROUP

*This expression was coined by Paul Reich III and Fred Ford, the two most clever game programmers who ever lived, in reference to Star Control 2, which was the most fun and ingenious game ever written. Word!

REVIEW

By Bradley Kaldahl and Jacob Kirk

Macromedia Flash; Visual vs. Language-Based programming

Macromedia Flash has taken the web by storm and with it a new programming paradigm has evolved that we refer to as "Visual Programming." Non-traditional "Visual Programmers" are producing algorithms that are really quite intriguing. In some respects this new visual paradigm represents an opportunity to create algorithms based on visual logic, which is an interesting alternative to traditional language-based algorithms. Please read on to see if you agree.

In the traditional language-based programming paradigm, the developer relies almost completely on the programming code to accomplish the desired task. While several highly respected authors such as Grady Booch and Coad & Yourdon have created visual diagram-based models for OO development, ultimately the design is translated into code. On the other hand Flash allows both design and implementation in a visual environment. This visual style of programming obviously has a strong appeal to certain individuals, most notably creatives, designers, and those unfamiliar with traditional programming languages and conventions.

Any programmer called upon to work or consult on an ActionScript project should be aware of the issues and idiosyncrasies of this new trend in "Visual Programming" as they will likely encounter legacy code that has been developed using the visual paradigm.

Problem definition

To contrast the visual paradigm with the language-based approach we have defined a simple programming problem.

Using Macromedia Flash 5, animate an object to move across the stage. When the user rolls-over the object have it reverse direction.

If you are familiar with Flash and ActionScript then we hope that you will enjoy the contrast of these two disparate techniques. If you are new to ActionScript we hope you will benefit from our discussion of some of the programming anomalies you will encounter in Flash when using the language-based approach.

Anomalies in ActionScript

While ActionScript resembles JavaScript, it is helpful to remember that ActionScript must conform to the animation and encapsulation features found in Flash. One of the more noticeable programming differences you will encounter in Flash is that you cannot animate an object using a standard FOR or WHILE repeat loop. There has been some debate as to why this is the case. Some have suggested that it is because the computer can execute a repeat loop at a much faster rate than the screen can redraw. While on the surface this sounds plausible, it suggests that computer animation within a repeat loop cannot be accomplished. According to Michael Williams, Associate Product Manager at Macromedia: "The issue here is not the fact that FOR and WHILE execute too quickly, it is because ActionScript, on a frame, executes before the stage is updated. The stage is only updated after all scripts on that frame have been executed."

If you do attempt to use the FOR or WHILE repeat loop for animation you will most likely encounter the following alert.

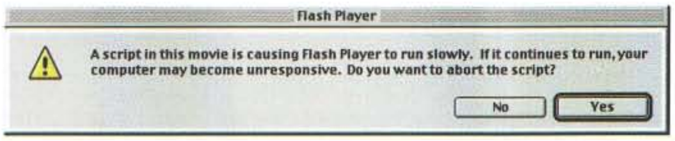

```
Flash Player
A script in this movie is causing Flash Player to run slowly. If it continues to run, your computer may become unresponsive. Do you want to abort the script?
```

***Figure 1**. Flash places a 15 second time-out limit on FOR and WHILE repeat loops.*

Bradley Kaldahl is a professor of Computer Publishing at Montgomery College and is author of the book EZ Flash 5. You can contact him through www.hotFlashBook.com

Jacob Kirk is a Designer, Teacher, Author and an all around Interactive Web "Whiz-kid". He specializes in using Macromedia Flash and Adobe LiveMotion. More information about what Jacob is up to can be found at www.EmazingMedia.com.

Mr. Williams explains: "The Flash Player itself provides this option to abort the script. The Flash Player will do this when it plays a movie which contains a one-frame ActionScript loop which can repeat itself indefinitely or for too long. A movie with such a loop can tax the resources of the computer system playing it. Giving viewers the option to end the script helps the Flash Player prevent an SWF file from freezing the viewer's browser or system. To avoid creating a movie that triggers this error message when it is played in the Flash Player, create a loop that goes through frames. You can do this by evaluating a variable in an if statement and, if the variable conditionalizes as true, then have the movie break the loop by going to the next frame. If the variable does not conditionalize as true, then have the movie loop back to the first frame (usually two to three frames back) of the movie."

The question still remains: How can an object be animated and controlled with ActionScript?

As discussed by Mr. Williams: "Use ActionScript applied to a Movie Clip, also known as clipEvents. For example, you have a movieClip on stage named "circle" the following code will move the circle across the X axis 10 pixels once per frame:

```
onClipEvent(enterFrame) {
     _x += 10;
}
```

This is dependent on the frame rate of the movie. In the default case this will refresh 12 frames/sec, but this can be increased or decreased by changing the movies frame rate at authortime."

The Language-Based Solution

PsuedoCode

```
Move the object by 15 pixels per frame.
   If the object goes off the left edge of the stage
      then place the object on the right side of the stage.
   If the object goes off the right side of the stage
      then place the object on the left side of the stage.

When the user rolls-over the object
   stop the object from moving.
When the user rolls-off the object
   reverse the direction of the object.
```

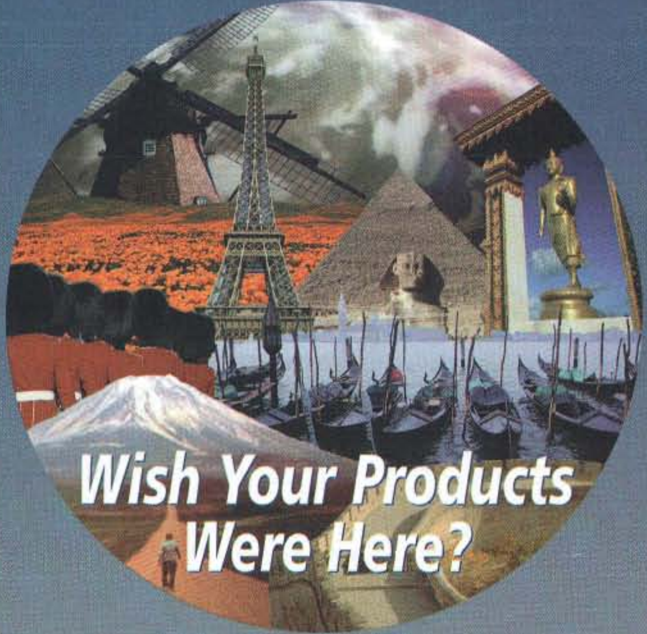
Listing 1: Language-Based MovieClip Script

Animate and wrap object.

/* onClipEvent(load) is being used to define some of the variables.
The variable moveAmount is the number of pixels the object will be animated.
ballLoc: (ball location) is used to define the location of the center point of the movieClip that encapsulates the ball-Button object.
_parent: is a relative reference to the movieClip that encapsulates the ball object.*/

```
onClipEvent (load) {
  moveAmount = 15;
  ballLoc = _parent

}
```

Classic Or Cocoa Applications? We Localize Them All!



- Translation
- Engineering
- Localization
- Project Management
- Desktop Publishing
- Quality Assurance

To receive your FREE copy of **The Guide to Translation and Localization – Preparing Products for the Global Marketplace:**

Call: **1-800-878-8523**
Fax: **1-503-419-4873**
Email: **info@lingosys.com**
Or visit: **www.lingosys.com**

lingo SYSTEMS

15115 SW Sequoia Pkwy. #200 Portland, OR 97224

```
/* The (enterFrame) clipEvent is the animation repeat loop used by Flash.
ballLoc._x += moveAmount will move the object by the moveAmount each
time a new frame is called. The two IF statements (with the magic numbers)
are checking to see if the object has exceeded the left or right edge of the
stage. They are set to create an object wrap so the object appears to leave
one edge and return on the opposite edge of the stage. */

onClipEvent (enterFrame) {
  ballLoc._x += moveAmount;
  if (ballLoc._x>640) {
    ballLoc._x = 0;
  }
  if (ballLoc._x<0) {
    ballLoc._x = 640;
  }
}
```

Listing 2: Language-Based Object Script

Reverse object direction.

```
/* This script is placed inside a button object on the stage.
Both rollOver and rollOut use the this. reference which insures that only the
object that received the event will have its variables altered. If multiple
instances of the ball object were used without the this. parameter all the
objects on the stage would reverse direction. storePosition is a global
variable created on the fly. It holds the original moveAmount before
temporarily setting the moveAmount to 0 to stop the object. rollOut
restores the original moveAmount, then reverses the direction of the object
by setting it to the opposite of its original value. */

  on (rollOver) {
    this.storePosition=moveAmount;
    this.moveAmount=0;
    }
  on (rollOut) {
    this.moveAmount = storePosition;
    this.moveAmount = moveAmount *-1;
    }
```

Since the programmer has taken control of the object with ActionScript, knowledge of the timeline is not needed. This animation takes place on a single frame. However, the developer does need to know how to create a button symbol since it is the only object that will respond to a rollOver event. Additionally the developer needs to know how to encapsulate the button in a movieClip symbol, since it is the only object that will accept a clipEvent.

In contrast the visual solution, presented below, utilizes the Flash timeline and its tweening capabilities to create the movement. The animation capabilities of Flash simplify the scripting.

The Visual Solution

Some might refer to the visual solution as a "designers approach" or a "non-programmers" solution, but to minimize it would be inappropriate as it demonstrates many of the features that programmers strive for when creating an algorithm. It is simple, elegant, fast and responsive, and with Flashs built-in encapsulation capabilities, it is easily reusable. Bradley Kaldahl refers to the visual algorithm presented below as "Equal Parallel Inverse Animation's," or EPI.

Description of the visual solution

```
Frame #        Frame #
1-----------40 *   100----------140**
 >>>>>>>>>>>>>>>   <<<<<<<<<<<<<<
Animate Left to Right  Animate right to left
  * = Script to loop back to frame 1
  ** = Script to loop back to frame 100
```

The object is animated to move from left to right (frame 1-40) using a motion tween. In frame 1 the object is on the LEFT side of the stage and in frame 40 the object is on the far RIGHT. When the animation playback-head reaches frame 40 it loops back to frame 1 using a simple "gotoAndPlay" navigation script placed on the timeline. An equal inverse animation is created further out on the timeline, in which the object is animated from right to left (frame 100-140). In frame 100 the object is on the RIGHT side of the stage and in frame 140 the object is on the LEFT. The two animation's are on the same timeline but move in opposite directions as shown in **Figure 2**.

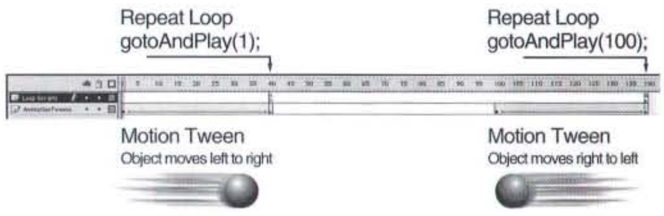


Figure 2, The Animation Timeline

When the animation begins, the playback head is looping from frame 1-40, i.e., the object is moving from left to right. To capture the rollOver event and jump the playback-head to the inverse animation another simple navigation script is placed inside the object.

```
On RollOver
gotoAndPlay frame (x of the inverse animation)
End RollOver
```

It is important to note that there are two separate instances of the ball object shown in figure 2 above. Each instance of the ball object can have a different script. The first instance begins on frame 1, and the second instance begins on frame 100. (Technically speaking there are actually 4 instances of the ball because each keyframe on the timeline represents an instance. Frame 40 and frame 140 both display a one frame instance of the ball object. For the sake of discussion we will refer to the first animation as instance one and the second animation as instance two). To view the code in the first instance of the ball object you would click on frame 1 on the timeline then click on the ball object on the stage. To view the code in the second

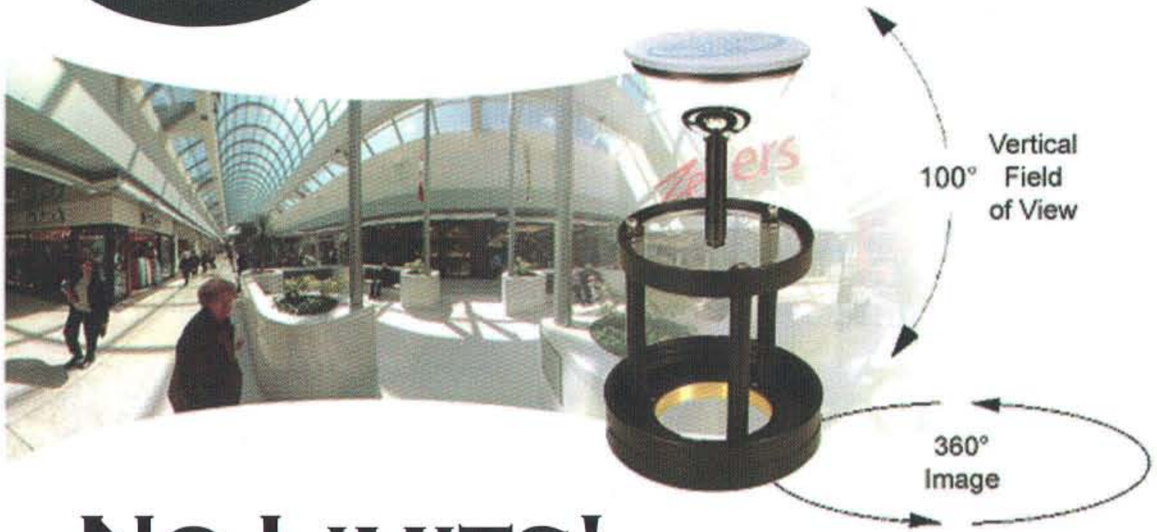
360 ONE VR

Macworld BEST OF SHOW 2 S 0 ★ 0 F 2

The 360 One VR™ is an amazing product that captures a complete 360° panoramic image in a single camera shot. The system is the perfect solution for capturing all the action, as it happens! Years of extensive research have produced an innovative solution that creates an immersive image without the restrictions or compromises normally associated with panoramic photography.

The system consists of a lightweight and rugged proprietary optical device and the innovative PhotoWarp™ software from EyeSee 360. The unique mirrored optic provides a complete 360° horizontal coverage with an outstanding 100° vertical field-of-view (50° above and 50° below the horizon).

ONE SHOT



The optic has a standard 67mm thread mount and can be adapted to a number of cameras using either Kaidan camera specific mounting kits or commercially available step-up rings that fit your camera. For a complete list of supported cameras and kits, please visit the 360 One VR web site.

NO LIMITS!

The 360 One VR is available with a variety of photographic accessories. These include: monopods, tripods, bubble levels, camera bags and Pelican hard cases. You can purchase them separately or together at a discount in either our Pro Accessory or Real Estate Kit for the 360 One VR.

For more information on camera mounting kits and all the latest accessories, visit our website.

KAIDAN http://www.360OneVR.com

QuickTime

Built for Mac OS X

EyeSee360°

Kaidan Incorporated
703 East Pennsylvania Blvd. Feasterville, PA 19053, USA
Phone: 215-364-1778 • Fax: 215-322-4186 • info@kaidan.com • http://www.kaidan.com

http://www.eyesee360.com

instance of the ball object, click on frame 100 on the timeline then click on the ball object on the stage.

The final question is: Which frame in the inverse animation will produce a smooth reverse?

Algorithm - Equal Parallel Inverse Animation's with Flash

Flash provides an easy-to-use property called `_currentFrame`, which provides the number of the frame when the event occurs. Using this property the following algorithm finds the inverse frame in the reverse animation.

> Inverse of first the first animation instance = Number of Frames in the Animation - Current Frame Number + Distance to the Reverse Animation or (NFA - CFN + D2RA = Inverse).
>
> Inverse of the second animation instance = Distance to the Reverse Animation - Current Frame Number + Number of Frames in the Animation or (D2RA - CFN + NFA = Inverse).

For example, using figure 2 and discussion above, if the user rolled over the object while it was on frame 23... NFA(40) - CFN(23) + D2RA(100) = Frame number 117.

By jumping the playback head to frame 117 the object appears unchanged with the exception of reversing direction of movement.

Listing 1: Visual Solution, first animation timeline loop script

Placed in frame 40 of the timeline

```
gotoAndPlay (1);
```

Listing 2: Visual Solution, second animation timeline loop script

Placed in frame 140 of the timeline

```
gotoAndPlay (100);
```

Listing 3: Visual Solution, first object instance script

Placed in the first instance of the object

```
//The Number of Frames in the Animation = 40;
// Distance to the Reverse Animation = 100;

on (rollOver) {
  stop ();
}
on (rollOut) {
  gotoAndPlay (40 - _currentFrame + 100 );
}
```

Listing 4: Visual Solution, second object instance script

Placed in the second instance of the object

```
on (rollOver) {
  stop ();
}
on (rollOut) {
  gotoAndPlay (100 - _currentframe + 40 );
}
```

Encapsulation

Encapsulation is what really makes Flash so powerful. Converting either the language or visual animation into a movieClip not only captures the animation but also encapsulates both Frame and Object scripts. Basically a movieClip becomes a self contained animated object that knows how it should behave to specific events. Once encapsulated as a movieClip multiple copies can be dropped on the stage and each plays independently. We have encapsulated and extended the code from this article to produce a couple of different web games. The games and source code can be found at www.hotFlashBook.com

Conclusion

While certain problems clearly require a language-based solution, such as score keeping in a game, a variety of other functions are more easily accomplished using the visual paradigm. To see two examples that appear fairly complex but were easily accomplished using the visual solution visit the web site listed below. The fact that these functions can be accomplished with little code and an aptitude for logical thinking makes it easy to see why many creatives are so excited about Flash. It is interesting that, when presented with a Flash problem, we both tend to look for code-based solutions first, yet on several occasions we have been delighted to discover simpler visual solutions that met our needs. One thing that is clear is that ActionScript programmers need to be familiar with Flash and its visual capabilities as well as the scripting language. An excellent resource for both novice and experienced Flash users is the book EZ Flash 5 by Bradley Kaldahl. It uses short 2-8 page hands-on projects to provide a wealth of both visual and code-based ideas and solutions. To find out more about the book EZ Flash 5 visit www.hotFlashbook.com

Acknowledgements

Special thanks to **Michael Williams**, from Macromedia, for your insight and information and to **Suzy Ramirez**, from Macromedia, for your wonderful assistance.

Multiple formats. Multiple platforms. Complex installers.

Aladdin solves the compression and installation puzzle.

Trying to figure out how to handle multiple compression formats and platforms?
The StuffIt Engine solves the compression puzzle.

Aladdin's StuffIt Engine SDK:

- Adds value to your application by integrating powerful compression and encryption.
- Is the only tool that supports the StuffIt file format.
- Provides a single API that supports over 20 compression and encoding formats common on Macintosh, Windows, and Unix.
- Makes self-extracting archives for either Macintosh or Windows.
- Available for Macintosh, Windows, Linux, or Solaris.

Licenses start as low as $99/year

To learn more, visit: www.stuffit.com/sdk/

StuffIt Engine SDK™

The power of StuffIt in your software.

Looking for the easiest and fastest way to build an installer?
StuffIt InstallerMaker completes your puzzle.

It's not enough just to write solid code anymore. You still have to write an installer for your users. StuffIt InstallerMaker makes it simple and effective.

- StuffIt InstallerMaker gives you all the tools you need to install, uninstall, resource-compress or update your software in one complete, easy-to-use package.
- Add marketing muscle to your installers by customizing your electronic registration form to include surveys and special offers.
- Make demoware in minutes. Create Macintosh OS X and Macintosh Classic compatible installers with StuffIt InstallerMaker.

Prices start at $250

To learn more, visit: www.stuffit.com/installermaker/

StuffIt InstallerMaker™

The complete installation solution.™

Aladdin Systems

www.stuffit.com
(831) 761-6200

PALM PROGRAMMING

By Andrew S. Downs

Web Clipping

Downloading a Palm OS Web Clipping Application via a Servlet

INTRODUCTION TO WEB CLIPPING

Several years ago the Palm VII introduced users to the wireless web through its built-in browser (Web Clipper, or simply Clipper). This browser supports Web Clipping Applications (WCAs), thin-clients built using Palm's supported subset of HTML. WCAs may contain pages consisting of controls, text, images and links.

A WCA is a specialized type of Palm OS database (as are other Palm OS apps). You construct a WCA using Palm's builder tool. Note that, although a hierarchy of directories and files may be used when building the WCA, inside the database everything exists in one logical directory (retaining the original file names), thus requiring unique names across all input files (both HTML and images).

An advantage to this database-oriented approach is that all of the elements necessary to display a set of pages may be assembled into one package for download and subsequent browsing. A continuous network connection is not required for viewing the pages, although it is possible (and common) to place live links on those pages.

The WCA discussed later in this article contains no live external links (see **Figure 1**). One benefit of this approach is that no additional web access is required when constructing or using the WCA, saving both time and money. (The Palm VII wireless network can be slow and expensive, which becomes an important factor if you need to build a WCA dynamically or access live data.) The application described here functions the same over a wireline modem, which may be more cost effective than a wireless connection.

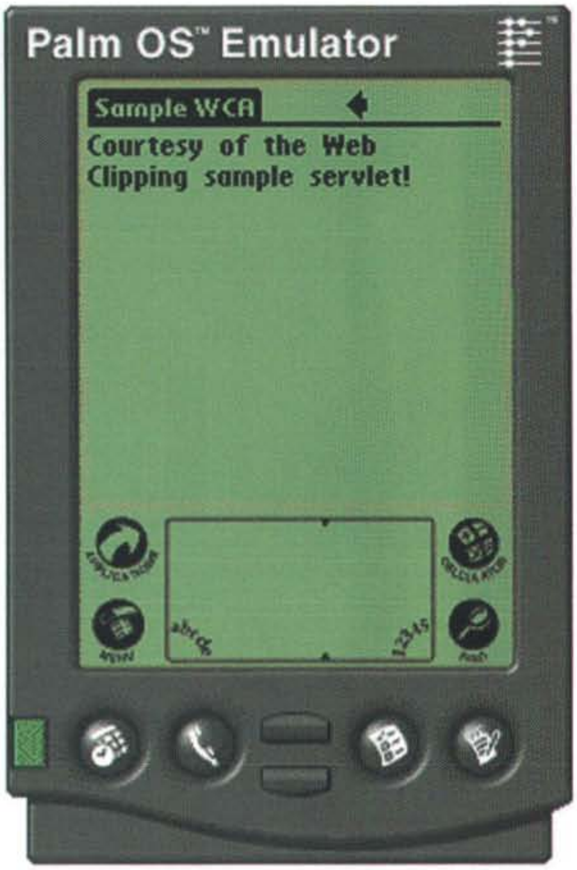


Figure 1. *The WCA running under Clipper in the Palm emulator.*

Non-Palm VII owners: Web Clipper is also included in the Mobile Internet Kit, a software upgrade that allows other Palm devices to access the Internet.

Andrew has worked with Palm OS since 1999. He wrote the Palm OS wireless client for Snippets Software. You can reach him at andrew@downs.ws.

InstallAnywhere
Mac OS X Edition
All the power to install and configure
Mac OS X software perfectly, every time.
The Industrial Strength Installer From
ZERO G

New OS. New Software. New Installer.

Introducing InstallAnywhere - Mac OS X Edition, the new power tool for your Mac OS X software installation. Stop using yesterday's tired old installer tools. Now, you can create industrial strength installers that are flexible, intuitive, and royalty-free. Your software will look better than ever and install perfectly, every time.

InstallAnywhere supports all Mac OS X features, like user authentication, file permissions, installing icons into the Dock, and offers a fully customizable Aqua look and feel.

Software innovators like Adobe, Apple, Borland, Gracion, LimeWire, and Sun already depend on Zero G for their software installation needs. See for yourself why InstallAnywhere - Mac OS X Edition is the new power tool for your software.

InstallAnywhere - The Industrial Strength Installer From Zero G.

Download a free trial version from http://www.ZeroG.com/goto/mac

zero ZERO G

InstallAnywhere Mac OS X Edition

X Built for Mac OS X

Components

There are several components to the system described in this article:

- A native Palm app (a .prc) that initiates the request to download the WCA. This app contains information needed to connect to a server, including URL and username.
- A Java servlet that returns the WCA from the server.
- The raw material for the WCA. In this example, it is a static text page.
- The Palm Query Application Builder (QAB) that creates the WCA. This tool is freely available for download. As of this writing the Macintosh version does not support execution via AppleScript or otherwise allow for command-line building or the inclusion of params. As always, check the Palm website for updates.

Palm App

The majority of the code presented in this article is for the Palm OS client application. The core functionality is in the Connection class (C++). It sequences the calls to retrieve and display a WCA.

Listing 1: Connection.cpp

Connection

```
The methods contained here include:
  Init: drives the overall connection and download process.
  Connect: sets up portions of the http request, and hands-off to a library-specific
    method.
  LaunchClipper: invokes WebClipper to display the WCA. Most of this method came
    from Palm's website.
  InvokeINetLib: use the INetLib to connect to a remote URL and download data.

  const int kGetFileSize = 0;
  const int kGetFile = 1;
  const ::Char * kUser = "sample";
  const ::Char * kUrl =
    "http://yourdomain:8080/servlet/WcaServlet";
  const ::Char * kDatabaseName = "Sample.pqa";
  const int kInBufferSize = 1024;
  const int kDefaultMsgSize = 1024;

void Connection::Init( void ) {
  // Although not an absolute requirement for a small WCA, if we know how much
  // space the downloaded WCA will take up, our code works more efficiently if we
  // only allocate a temp buffer of the necessary size. Since the Connect() method does
  // not currently return anything, there is code in InvokeINetLib() that saves the size.
  // That is not done in this method.
  Connect( kGetFileSize, kUser, kUrl );

  // Retrieve the actual WCA from the server.
  Connect( kGetFile, kUser, kUrl );

  // Display the downloaded WCA without additional user intervention.
  LaunchClipper( "file:Sample.pqa" );
}

void Connection::Connect( int op, ::Char * user,
  ::Char * url ) {
  // This string holds the additional params in the http request.
  ::Char * buf;

  // The string representation of the operation code goes here.
  ::Char opString[ 2 ];

  // Allocate a buffer for our outgoing request. The default size is stored in another class.
  buf = ( ::Char * )::MemPtrNew( kDefaultMsgSize );

  if ( buf ) {
    // Initialize the string.
    ::MemSet( buf, kDefaultMsgSize, '\0' );
    ::MemSet( opString, 2, '\0' );

    // Set the operation code.
    ::StrIToA( opString, op );

    // Create the interesting part of the request string, formatted for an http GET
    // method. Append the user and operation code params.
    ::StrCat( buf, "?user=" );
    ::StrCat( buf, user );
    ::StrCat( buf, "&op=" );
    ::StrCat( buf, opString );

    // Invoke a library-specific method to connect to the server.
    // If we are not only using InetLib then wrap this in a conditional.
    InvokeINetLib( url, buf, op );

    ::MemPtrFree( buf );
  }
}

// Most of this method came from Palm's website.
::Err Connection::LaunchClipper( const ::Char * origurl ) {
  ::Err err;
  ::Char * url = 0;
  ::DmSearchStateType searchState;
  ::UInt16 cardNo;
  ::LocalID dbID;
  ::UInt16 length = ::StrLen( origurl );

  // Copy the URL, since the OS will free the parameter once Clipper quits.
  url = ( ::Char * )::MemPtrNew( length );

  if ( !url )
    return sysErrNoFreeRAM;

  ::StrCopy( url, ( const ::Char * )origurl );

  ::MemPtrSetOwner( url, 0 );

  // Locate and launch Clipper.
  err = ::DmGetNextDatabaseByTypeCreator( true,
    &searchState, sysFileTApplication, sysFileCClipper,
    true, &cardNo, &dbID );

  // If Clipper is not present...
  if ( err ) {
    ::FrmAlert( NoClipperAlert );
    ::MemPtrFree( url );
  }
  else {
    err = ::SysUIAppSwitch( cardNo, dbID,
      sysAppLaunchCmdGoToURL, url );
  }

  return err;
}

long Connection::InvokeINetLib( ::Char *theURL,
  ::Char *theSuffix, int selector ) {
  long retval = -1;
  static int inBufferSize = 0;

  // Setup buffers.
  ::Char * in = (::Char *)::MemPtrNew( kInBufferSize );

  ::Char * out =
    ( ::Char * )::MemPtrNew( kDefaultMsgSize );

  // After this, we have a url in the buffer similar to:
  // http://www.yourdomain:8080/WcaServlet?user=sample& op=0
  ::StrCopy( out, theURL );
  ::StrCat( out, theSuffix );

  ::Err err;

  ::UInt16 libRefnum;

  // Load net library.
  err = ::SysLibFind( "INet.lib", &libRefnum );

  if ( err ) {
    ErrNonFatalDisplay( "Unable to find INetLib" );
    goto close;
  }
```

```
::MemHandle inetH;
::UInt16 indexP;
::INetConfigNameType config;

// Other possible values include inetCfgNameCTPWireless and
// inetCfgNameDefWireless.
::StrCopy( config.name, inetCfgNameCTPDefault );

// Get the configuration index of the net library.
err = ::INetLibConfigIndexFromName( libRefnum, &config,
  &indexP );

// Open the net library.
err = ::INetLibOpen( libRefnum, indexP, 0, NULL, 0,
  &inetH );

// Minor adjustments to the INetLib settings.
// Set the buffer size.
long tempValue = kInBufferSize;
::INetLibSettingSet( libRefnum, inetH,
  inetSettingMaxRspSize, &tempValue,
  sizeof( tempValue ) );

// Disable compression.
tempValue = ctpConvNone;
::INetLibSettingSet(libRefnum, inetH,
  inetSettingConvAlgorithm, &tempValue,
  sizeof(tempValue));

::MemHandle theSocket;

// So we don't wait forever...
::Int32 timeout = ::SysTicksPerSecond() * 15;

// Send our request to the URL specified in our output buffer.
err = ::INetLibURLOpen( libRefnum, inetH,
  ( unsigned char * )out, NULL, &theSocket, timeout,
  inetOpenURLFlagForceEncOff );

::UInt32 bytes = 0, tempBytes = 0;

::UInt16 status = 0;

::INetEventType event;

bool ready = false;

// Wait for a change in the socket's status, which will be the signal that there is a
// response to process.
while ( !ready ) {
  ::INetLibGetEvent( libRefnum, inetH, &event, timeout );

  if ( event.eType == inetSockReadyEvent ||
      event.eType == inetSockStatusChangeEvent )
    ready = true;
}

::Int32 inMaxBufferSize = kInBufferSize;

if ( selector == kGetFile && inBufferSize != -1 )
  inMaxBufferSize = inBufferSize;

::UInt32 numBytes = kDefaultMsgSize;

// The value of in will change as data gets read into the buffer, so save the original
// address for later.
::Char * oldIn = in;

// Read incoming data, looping while there is still data available and we have not
// downloaded the entire WCA (when applicable.)
do {
  tempBytes = 0;

  if ( ( inMaxBufferSize - bytes ) < kDefaultMsgSize ) {
    numBytes = inMaxBufferSize - bytes;
  }

  err = INetLibSockRead( libRefnum, theSocket, in,
    numBytes, &tempBytes, timeout );

  // Advance pointer.
  in += tempBytes;

  // Increment byte count.
  bytes += tempBytes;
```

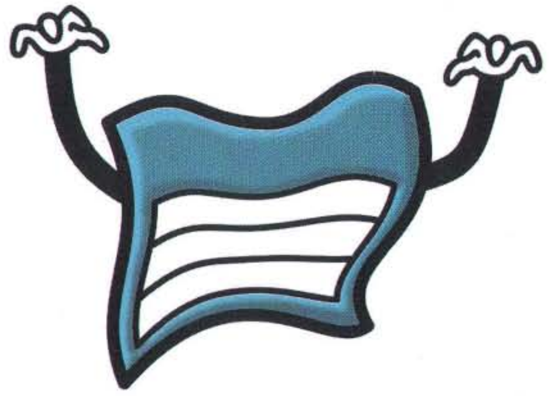
Tri-platform Calendaring

OS X *Carbon*
Mac OS *7.1 - 9.x*
Windows *95 - 2000*

New!

CalendarMonster 1.2

Multi-platform calendar



- **End-user Calendar**
  The best FMP calendar experience
  - Fast, fast, fast!!!
  - Platform-appropriate user experience
  - Highly scalable
- **Excellent Developer Support**
  Quickly and easily integrate with your FileMaker Pro projects
  - 100% live data, 100% modular, no plug-ins
  - FileMaker Pro 4 or 5 (including 5.5)
  - As little as 2 ScriptMaker calls
  - Dedicated developers' utility

only $49! per user

Developer and volume licensing available

ASCENDING TECHNOLOGIES

monster@asctech.com

Download your own FREE fully-functional demo

http://www.asctech.com

```
} while ( ( tempBytes != 0 ) &&
  ( bytes < inMaxBufferSize ) && ( !err ) );

// Close socket.
err = ::INetLibSockClose( libRefnum, theSocket );

// Restore pointer to incoming data.
in = oldIn;

if (selector == kGetFileSize && bytes > 0) {
  inBufferSize = ::StrAToI(in);
}

if (selector == kGetFile && bytes > 0) {
  // The expected WCA name should be in the stream.
  ::Char * dataP = ::StrStr( in, kDatabaseName );

  if ( dataP == NULL ) {
    ErrDisplay( "String not found" );
    goto close;
  }

  Int32 num = bytes - ( dataP - in );

  ::LocalID id = ::DmFindDatabase( 0, kDatabaseName );

  if ( id != 0 ) {
    err = ::DmDeleteDatabase( 0, id );

    if ( err != errNone )
      goto close;
  }

  // We will first write the raw data to a temporary database.
  // Check whether that database already exists (a bad thing).
  id = ::DmFindDatabase( 0, "tempSample" );

  // If we did not clean up previously, delete the temp database.
  if ( id != 0 ) {
    err = ::DmDeleteDatabase( 0, id );

    if ( err != errNone )
      goto close;
  }

  // Create a temp database, assigning Clipper as the owner.
  err = ::DmCreateDatabase( 0, "tempSample", 0x636c7072,
        0x70716120, true );

  // Note: from here to the end of the method some of the error checking has been
  // relaxed in order to shorten this example. Production code should check every
  // return value and take appropriate action.
  if ( err != errNone )
    ErrDisplay( "DmCreateDatabase() returned err !=
      ErrNone");

  // Ensure that our creation attempt succeeded.
  id = ::DmFindDatabase( 0, "tempSample" );

  if ( id == 0 )
    ErrDisplay( "DmFindDatabase() returned id == 0" );

  // Open the database for writing.
  ::DmOpenRef ref = ::DmOpenDatabase( 0, id,
    dmModeReadWrite );

  if (ref == 0)
    ErrDisplay( "Error opening database" );

  // Create a resource of type 'pqa '.
  ::MemHandle res = ::DmNewResource( ref, 0x70716120, 0,
    num );

  if ( res == NULL )
    ErrDisplay( "DmNewResource() returned NULL" );

  // Lock the resource for use.
  ::MemPtr ptr = ::MemHandleLock( res );

  if ( ptr == 0 )
    ErrDisplay( "MemHandleLock() returned 0" );

  // Write the resource into the database.
  err = ::DmWrite( ptr, 0, dataP, num );

  if ( err != errNone )
    ErrDisplay( "Error writing resource" );

  // Use the raw data to create the "real" WCA. Not condoned by Palm
  // for non-system databases.
  err = ::DmCreateDatabaseFromImage( ptr );

  // Unlock and free up memory.
  err = ::MemHandleUnlock( res );

  if ( err != 0 )
    ErrDisplay( "MemHandleUnlock() returned err != 0" );

  err = ::DmReleaseResource( res );

  // At this point we are so close to being done that success is likely.
  // Still, for consistency we check result codes.
  if ( err != errNone )
    ErrDisplay( "DmReleaseResource() returned
      err != errNone" );

  err = ::DmCloseDatabase( ref );

  if ( err != errNone )
    ErrDisplay( "DmCloseDatabase() returned
      err != errNone" );

  // Remove the temporary database.
  err = ::DmDeleteDatabase( 0, id );

  // Locate the real database and check that we can open it for reading.
  id = ::DmFindDatabase( 0, kDatabaseName );

  if ( id != 0 ) {
    ref = ::DmOpenDatabase( 0, id, dmModeReadOnly );
    err = ::DmCloseDatabase( ref );
  }

  // Return the number of bytes read.
  retval = bytes;
}

close:
  err = ::INetLibClose( libRefnum, inetH );

  // Cleanup allocated memory.
  if ( out )
    ::MemPtrFree( out );

  // Reset pointer.
  in = oldIn;

  if ( in )
    ::MemPtrFree( in );

  return retval;
}
```

A Starter WCA

The Web Clipping Application in this example is intentionally simple. A WCA gets created using the Query Application Builder tool, from one or more HTML and image files. This example contains static text only, contained in one source HTML file. You can extend this WCA by adding a link or button that triggers a fetch of the latest information from the server.

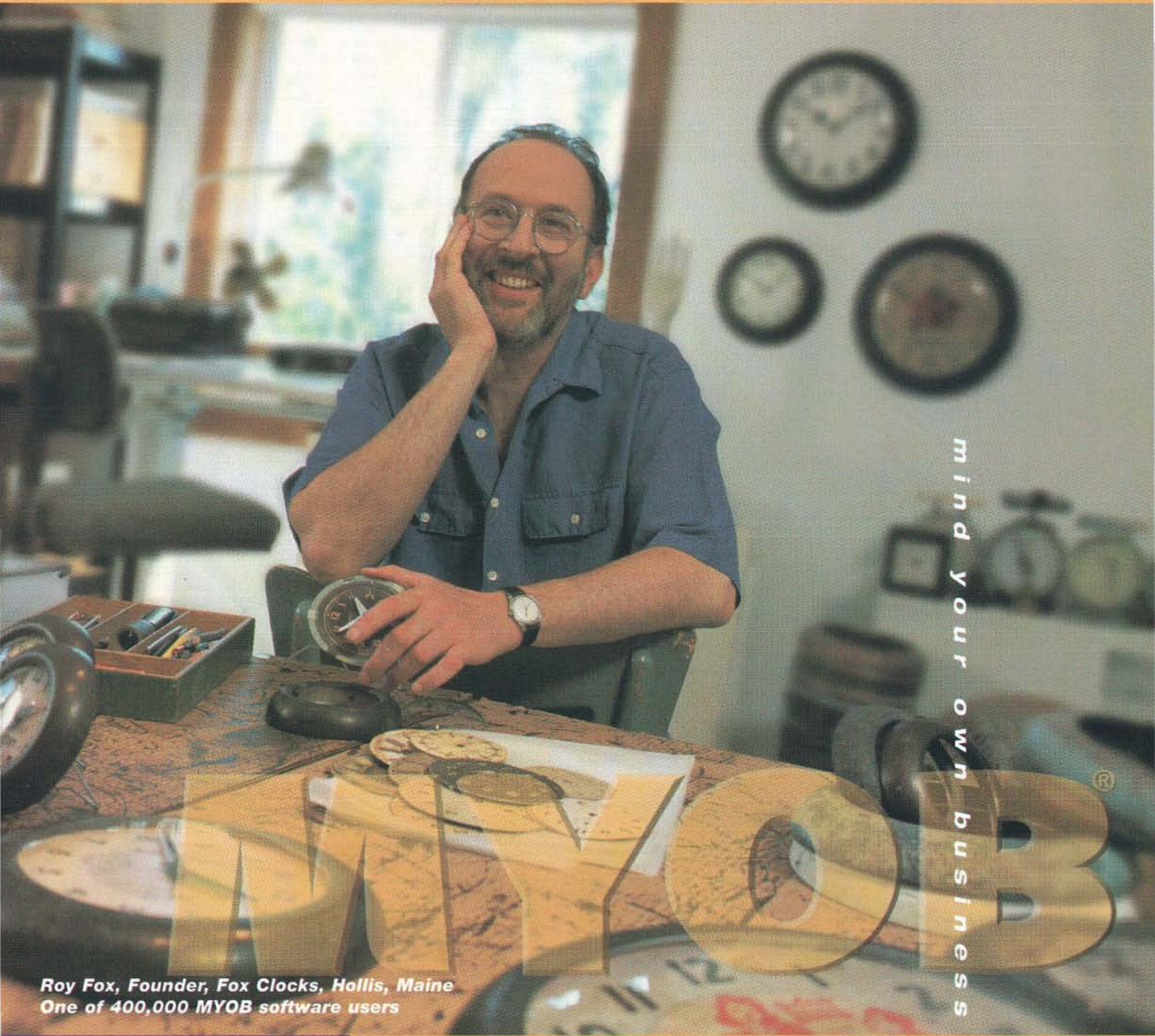
Listing 2: index.html

A starting point for a Web Clipping Application (WCA). Note the inclusion of the Palm identifier in the meta tag.

```
<html>
  <head>
    <meta name="palmcomputingplatform" content="true">
    <title>Sample WCA</title>
  </head>
```

DESIGNED FOR THE WAY YOU WORK

I'd rather create clocks than invoices.
If I wanted to keep books all day, I'd have been an accountant.



Roy Fox, Founder, Fox Clocks, Hollis, Maine
One of 400,000 MYOB software users

MYOB

Small Business.
Smart Solutions.℠

AccountEdge on the Mac.
Software includes QuickBooks® conversion utility. MYOB also offers conversion service.

800.322.MYOB
myob.com/us

®QuickBooks is a trademark of Intuit, Inc.

MYOB software is the simplest, most powerful, most complete solution for managing my company on the Mac, from the day to day to the bottom line.

Antique frames. Quartz movements. That's my business.
MYOB software works for me.

```
  <body>
    <h3>Courtesy of the Web Clipping sample servlet!</h3>
  </body>
</html>
```

The servlet

The Java servlet illustrated here responds to client http requests, and can run on something as simple as Sun's servletrunner application (found in older versions of the Servlet Development Kit.) This servlet's primary task is to return the (already-built) WCA associated with a particular id.

This servlet accepts two parameters in the http request:

- the name of a user, allowing us to return WCAs tailored to specific individuals, group, etc.
- an operation code, allowing for multiple tasks to occur. This servlet can return the size of the WCA as a separate operation. This allows a client to request the WCA size first, setup a buffer to hold the actual WCA, then request the WCA itself.

Listing 3: WcaServlet.java

WcaServlet.java

Receive http requests from clients and return a built WCA.

```
import java.io.*;
import java.net.*;

import javax.servlet.*;
import javax.servlet.http.*;

public class WcaServlet extends HttpServlet {
  // Elements passed in the request string.
  static final String kOperation = "op";
  static final String kUser = "user";

  // Operations we handle. Passed in request string.
  static final int kGetFileSize = 0;
  static final int kGetFile = 1;

  // Name of actual pqa file should be the same for all users. For flexibility, we can
  // retrieve it from different folders as needed.
  final String kPqaFilename = "Sample.pqa";

  // Name of base directory (relative to servlet dir) from which to build path to pqa.
  final String kPartialPath = "dev/pqa/";

  // Most servlets do most of their work starting from doGet() or doPost().
  public void doGet( HttpServletRequest request,
    HttpServletResponse response )
    throws ServletException, IOException {
    // We can handle different users. The request carries the username as a param.
    String user = request.getParameter( kUser );

    // The operation of interest also gets passed as a param.
    int op = Integer.parseInt(
   ( String )request.getParameter( kOperation ) );

    // The output stream is where our response will go.
    ServletOutputStream out = response.getOutputStream();

    switch ( op ) {
      // The file size is important when the receiver needs to know how big a buffer
      // to allocate for incoming data.
      case kGetFileSize:
        File pqa = new File( kPartialPath + username
          + "/" + kPqaFilename);

          // Write the file size to the output stream.
          if ( pqa != null && pqa.exists() && pqa.isFile())
            out.println( pqa.length() );

        break;

      // Return the actual pqa in the output stream.
      case kGetFile:
        // Build the path to the file, and attempt to open the file.
        FileInputStream fis = new FileInputStream(
          kPartialPath + user + "/" + kPqaFilename );

        if ( fis != null ) {
          // Open a "pipe" to get data out of the file.
          DataInputStream bis = new DataInputStream( fis );

          // Copy the pqa to the output stream. This particular implementation can
          // be improved upon by copying more than a byte at a time.
          while ( bis.available() > 0 )
            out.write( bis.readByte() );

          bis.close();
          fis.close();

          out.close();
        }

        break;

      default:
        break;
    }
  }
}
```

It is possible to extend this servlet to dynamically build a WCA on demand. This would allow up-to-the-minute data to be inserted into a WCA targeted at a particular user. The operation code allows for some sophisticated handling of such user requests. For example, the servlet could respond to an initial request for a WCA by spawning a thread to build that WCA dynamically. Since it takes time to perform such a build, particularly if data must be fetched from the Internet, it would be desirable to add some status codes to the process. A client can request the current status of the build, loop while it is not complete, then request the WCA itself.

Enhancing the system

There are many bells and whistles that can be incorporated into this system. For example, the Java servlet may connect to an LDAP server to validate the user and obtain user-specific configuration information. Several servlet engines are available that can run this servlet, including the servletrunner application from Sun (good for testing), and apache.org's Tomcat. Non-Palm OS devices may be supported by the servlet: the type of client requesting a file could be sent as a param in the http request.

Many WCAs consist of one or more statically coded pages. But dynamic pages often work much better if you have access to a reliable mechanism for generating those pages. Java servlets provide an easy way to gather pages, build a WCA via a command-line prompt, and then download the WCA to the Palm device.

An alternative to building the WCA using the Palm tools would be to write the database format directly. Although the format has probably remained static over the past two years, it requires time (and money) to implement such a writing mechanism, whereas the build tools can be downloaded and setup very quickly.

Bibliography

Combee, Ben and R. Eric Lyons, David C. Matthews and Rory Lysaght. Palm OS Web Application Developer's Guide. Syngress Publishing, Inc., 2001.

Bachmann, Glenn. Palm Programming. Sams Publishing, 1999.

Hunter, Jason and William Crawford. Java Servlet Programming. O'Reilly & Associates, Inc., 1998.

Holds up to 256MB and fits on your key chain or in the smallest pocket so you can transport your files anywhere!

USB Flash Key™

The first nonvolatile FireWire™ flash key chain storage solution. Supports CompactFlash™ and IBM MicroDrives™.

wiebeTECH

KeyChain™

portable storage

SMARTDISK
Simplifying The Digital Lifestyle

FireFly™

Holds up to 5GB of anything you choose! The FireWire™ connection allows for a transfer rate of 12MB per second.

wiebeTECH

MicroGB™

Has the capacity to hold up to 48GB of information and offers both FireWire™ and USB interface.

Dr. Bott LLC

www.drbott.com 877.611.2688

RESELLERS! Hundreds of Mac-friendly products available through Dr. Bott

by Bob Boonstra, Westford, MA

ENDGAME

The Chess players among you have another opportunity to excel this month. Back in 1995 (have I really been writing this column for that long?), we had a Challenge that required readers to identify all possible chess positions from a given board position. Gary Beith won that Challenge, and you might want to refer back to his solution when thinking about this month's problem, which invites you to solve the chess end game.

This month's problem is formulated as a C++ class. The prototype for the code you should write is:

```
typedef enum {
  kEmpty=0, kPawn, kKnight, kBishop, kRook, kQueen, kKing
} ChessPiece;

typedef enum {
  kNone=0,kWhite=1, kBlack
} Player;

typedef struct Square {
  ChessPiece piece;
  Player player;
} Square;

typedef struct PieceLocation {
  char row;
  char col;
} PieceLocation;

typedef Square Board[8][8];
  /* indexed as [row][col] */
  /* white start row is 0 */
  /* columns numbered left to right viewed from row 0 */

typedef struct Move {
  Board board;
    /* board before move is made */
  Player playerMoving;
    /* which player is making this move */
  PieceLocation pieceBeingMoved;
    /* which piece is being moved */
  PieceLocation destination;
    /* where is pieceBeingMoved being moved to */
  Move *alternativeMove;
    /* list pointer for alternative moves by playerMoving;
       NULL if no more alternatives */
  Move *subsequentMove;
    /* subsequent move in this move tree, made by opponent to playerMoving,
       NULL if no subsequent move is possible */
  bool moveIsCapture;
    /* true if this move is a capture */
  bool moveIsCheck;
    /* true if this move places the opponent in check (but not mate) */
  bool moveIsMate;
    /* true if this move mates the opponent */
  bool moveIsPromotion;
    /* true if this move promotes a pawn at the 8th rank */
  ChessPiece pieceAfterPromotion;
    /* valid only if moveIsPromotion is true, set to value of new piece */
} Move;

class EndGame {
  Board board;
  Move *initialMove;
  /* add other private members as you see fit */
  ~EndGame(void) {
    /* your destructor code goes here */
  }
public:
  EndGame(Board initialBoard, Player playerToMove) {
    /* your constructor code goes here */
  }
  Move *Solve() {
    /* your code goes here */
    return moveTree;
  }
};
```

The constructor for your EndGame class is provided with the initial board configuration (board) and the identity of the player who moves first (playerToMove). When your Solve method is called, your job is to choose an initial move that leads to checkmate in the minimum number of moves possible. You also need to compute the possible opposing moves with which the other player might respond, and your response to each of those moves, and so on, until each branch of the tree results in checkmate.

Either your constructor or the Solve routine should allocate storage for your first move (moveTree). The Move structure contains the board configuration prior to making the move, the identity of the player making the move, the location and destination of the piece being moved, and some boolean values indicating the results of the move. It also contains a pointer to the next ply of the move tree (subsequentMove), and a pointer to alternative moves in the current ply (alternativeMove). The former contains moves made by the opposing player in response to this move, while the latter contains other moves that could be made by the current player instead of this move. Together, they allow you to describe the entire move tree.

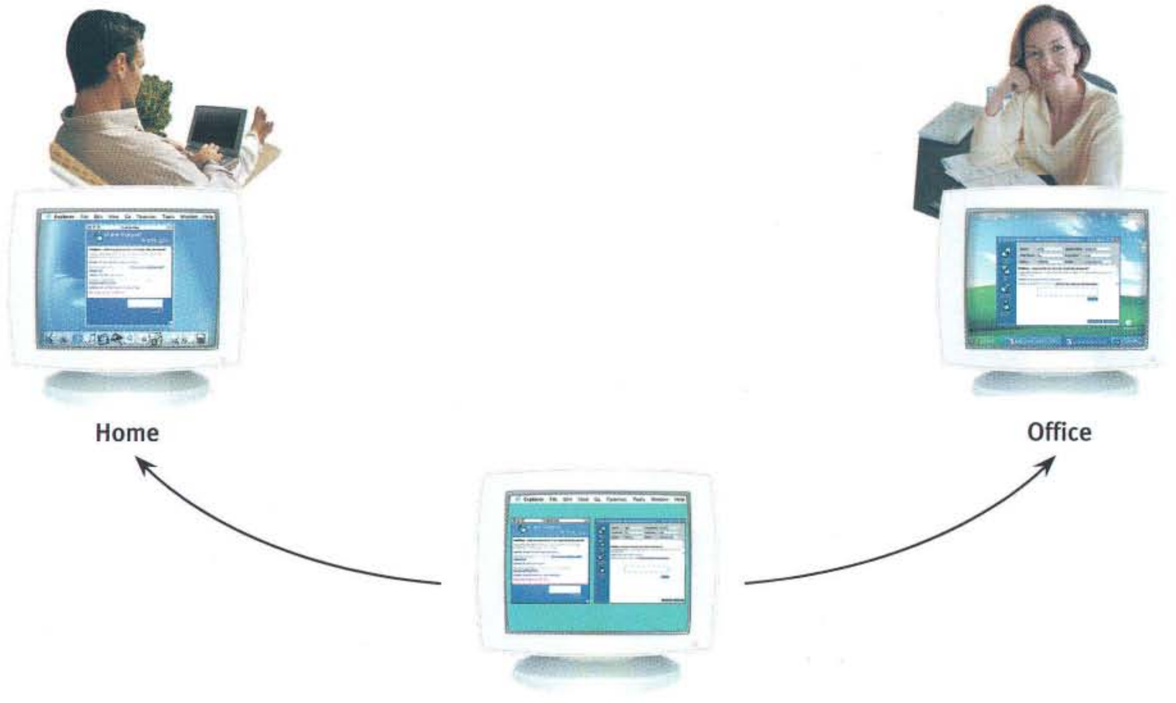
Your destructor needs to free any memory allocated for the Move data structure.

In calculating moves, you can assume that any castling that might take place has already done so. You can also ignore the possibility of en passant pawn moves. You do need to consider the possibility of promoting a pawn by moving it to the 8th rank of the board. The Move data structure makes provision for specifying the type of the promoted piece.

The winner of this Challenge will be the entry that correctly solves all test cases in the least amount of execution time. Correctness means identifying all moves in the move tree, and guaranteeing checkmate in the minimum number of moves. Timing will include the constructor, a call to your solve routine, and a call to your destructor for a sequence of boards. I am eliminating the subjective evaluation factor for this Challenge because of questions about fairness, but both readers and I appreciate code that is clear and well commented.

This will be a native PowerPC Carbon C++ Challenge, using the Metrowerks CodeWarrior Pro 7.0 development environment. Please be certain that your code is carbonized, as I may evaluate this Challenge using Mac OS X.

Yes, you *can* be in two places at once.



For more than a decade, Timbuktu Pro and netOctopus have been the leading remote control, file transfer and systems administration applications for the Mac OS. Now, both of these indispensable tools are updated to take full advantage of the world's most advanced operating system.

Timbuktu Pro

Whether you're at home or at work, Timbuktu Pro allows you to operate distant computers as if you were sitting in front of them, transfer files or folders quickly and easily, and communicate by instant message, text chat, or voice intercom.

http://www.timbuktupro.com

netOctopus

Intuitive and powerful, netOctopus can manage a network of ten or 10,000 computers. Inventory computers, software and devices on your network; distribute software; configure remote computers; and create custom reports on the fly.

http://www.netoctopus.com

Learn more, try it, or buy it online. Call us at **1-800-485-5741**.

timbuktu pro® • netOctopus

netopia.

Winner of the May, 2002 Challenge

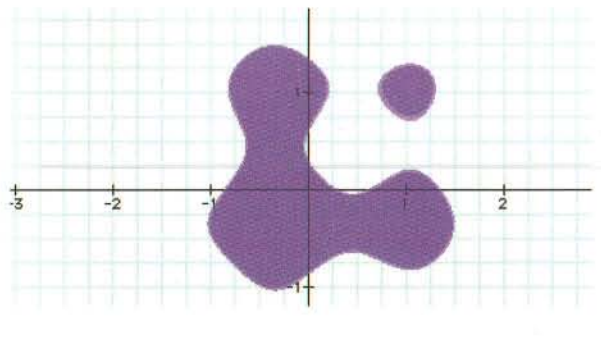
The May Challenge asked users to assemble a Jigsaw puzzle. The puzzle was presented as a color bitmap, with each piece of the puzzle consisting of contiguous pixels of a unique color. The pieces were disassembled and rotated by some multiple of 90 degrees. All of them were "face up". The puzzle was guaranteed to be rectangular in shape, and the pieces were guaranteed to fit together in only one way. The top left corner of the puzzle was in the correct position, which removed any ambiguity about orientation. Congratulations to **Tom Saxton** for submitting the winning entry for this Challenge.

Generating test data for the Challenge is often, well, a challenge, and this was particularly the case this month. I needed to ensure that the puzzle pieces I generated fit together without ambiguity. And while I could have generated puzzle pieces using "cuts" based on straight line segments, I wanted pieces that resembled real jigsaw puzzles. After struggling with this for some time, I nearly despaired of this objective, but eventually I found something that seems quite pleasing.

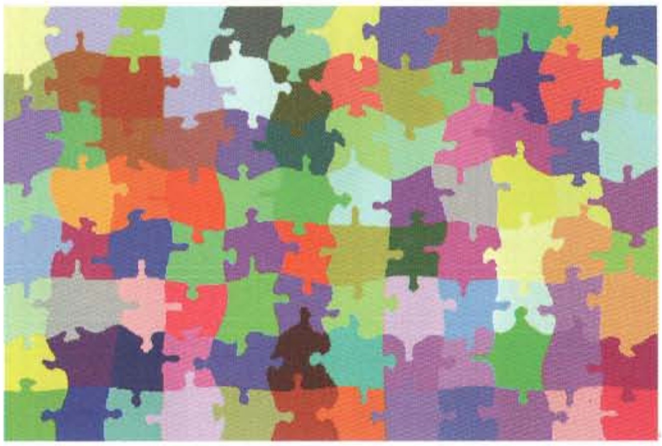
The first step in puzzle generation was to divide the puzzle into a roughly rectangular grid. For this, I decided to superimpose several cosine waves with random amplitudes, wavelengths, and offsets. I settled on summing four cosine functions for each horizontal or vertical "cut" in the puzzle, with amplitudes between 5% and 10% of the average piece size and wavelengths between 33% and ~300% of the average piece size. I repeatedly divided the puzzle into different colors for each of these horizontal and vertical cuts. This created a rectangular grid of uniquely shaped pieces with pleasantly curved edges.

I could have stopped there, but I really wanted to have pieces that had the protuberances or "ears" commonly found in real jigsaw puzzles. Being uncertain about how to create these using mathematics, I thought about creating ears manually and "stamping" them onto each edge, modifying them slightly to make them unique. That idea lasted all of about fifteen minutes, as tedium drove this thought from my mind. Eventually, I decided to experiment with one of my favorite programs, Graphing Calculator. An early version of GC came bundled with MacOS for some time, but <plug> the commercial version http://www.pacifict.com offers significantly more functionality </plug>. Anyway, GC has helped me out in the past, so I started experimenting, and ran across this sample equation:

■ $$(x^2 + y^2 + \sin 4x + \sin 4y) < 1$$



Looking at the part of this equation that is above the x axis with $-1<x<0.5$, it seemed pretty close to what I was looking for. And randomly varying the amplitude of the sin functions caused the ear to take different shapes. The coefficients in the sin functions needed to be constrained to between about 1.5 and 5.5 in order to prevent the ear from being pinched off and disconnected from the base. In my first few puzzles, a few of these disconnected pieces bypassed detection by me, but not by contestants, so I eventually settled on a more restricted range of parameters that generated slightly less interesting, but still acceptable, ear shapes. For each edge, I selected a location somewhere in the middle of the piece, randomized the direction of the ear (with a small probability of having no ear at all), superimposed the x axis above onto the curved line described above, and adjusted the piece boundary according to the portion of a shape like the one above located above the x-axis (or, for vertical lines, the portion right of the y-axis. This turned out to be less trivial than I had hoped, especially being careful to avoid creating those pesky disconnected pieces. Preventing one ear from intersecting with another ear and cutting a piece in two proved to be an additional complication, but the end result looked like this:



Tom locates the individual pieces of the puzzle in his _FFindPieces routine, and creates four bitmaps per piece, one for each possible rotation, in _FBuildPceBitmap. He detects edge pieces in the _GetEdgeInfo routine, marking them as such for future use. After placing the upper left piece in its guaranteed-correct position, the heavy lifting is done by the _FSolvePce routine, which solves the puzzle from top to bottom, left to right. Tom checks each possible rotation of each piece against the current piece location using two passes, the first trying to look at only pieces with the correct "innie/outie" (Tom's term) match, and the second to pick up pieces to weirdly shaped to be missed in the first pass. The logic for matching two pieces is rather intricate, and can be found in the _FTestPce routine.

Ernst solves the puzzles by first creating a vector array for each piece that described a clockwise path around the piece. He assembles the pieces by inserting them into a Shell data structure, first placing the edge pieces, and spiralling inward toward the center. While his solution is very fast, it became confused on the largest of my test cases, a problem that Ernst attributed to a lack of backtracking logic.

I evaluated the entries using 6 test cases that ranged in size from 24 pieces to 3750 pieces. Both Ernst and Tom were credited for including an optional feature to display the solved puzzle. Tom also displayed the puzzle in its disassembled state, and included options to display PICT and Jigsaw files, for which he earned a feature point reduction bonus. Ernst earned a larger point reduction for code clarity and commentary.

The table below lists, for each of the solutions submitted, the total execution time in seconds, the bonuses for clarity and for displaying the solved puzzle, and the total score. It also lists the programming language used for each entry. As usual, the number in parentheses after the entrant's name is the total number of Challenge points earned in all Challenges prior to this one.

	Time (secs)	Cases Correct	Clarity Bonus	Features Bonus	Score	Language
Tom Saxton (210)	597.1	6	0.10	0.25	388.1	C++
Ernst Munter (872)	89.9	5	0.25	0.20	49.5	C++

TOP CONTESTANTS ...

Listed here are the Top Contestants for the Programmer's Challenge, including everyone who has accumulated 20 or more points during the past two years. The numbers below include points awarded over the 24 most recent contests, including points earned by this month's entrants.

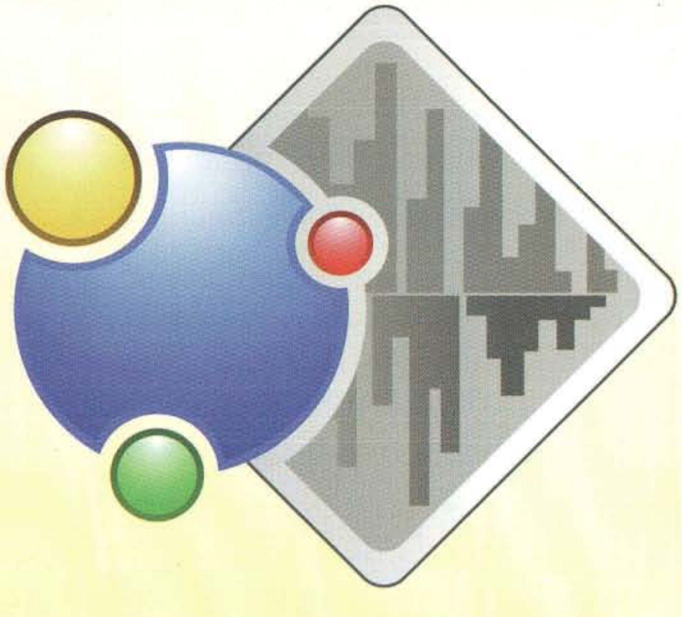
Rank	Name (24 mo)	Points (24 mo)	Wins Points	Total
1.	Munter, Ernst	243	8	872
2.	Saxton, Tom	65	2	230
3.	Taylor, Jonathan	57	2	83
4.	Stenger, Allen	53	1	118
5.	Rieken, Willeke	42	2	134
6.	Wihlborg, Claes	40	2	49
8.	Gregg, Xan	20	1	140
9.	Mallett, Jeff	20	1	114
10.	Cooper, Tony	20	1	20
11.	Truskier, Peter	20	1	20

Here is Tom's winning Jigsaw solution. The code had been abridged for publication because of page constraints; see http://www.mactech.com for the full version.

JIGSAW.CP

Copyright © 2002
Tom Saxton

```
/*
 * Jigsaw
 *
 * Created by Tom Saxton on Thu Apr 18 2002.
 */
```



IPNetMonitorX

- The Swiss Army Knife of Mac OS X Internet Diagnostic Tools
- 15 Fully Integrated Tools–Monitor, Link Rate, Address Scan, TCP Dump and more
- Troubleshoot and Solve Network Problems
- Responsive, Intuitive, Easy to Use Interface
- Fast Multi-Threaded Architecture–see network behavior as it happens
- 21-Day Fully Functional Free Trial

Only $40!

Download Your Copy Now at
www.sustworks.com

SUSTAINABLE Softworks
Tools for Internet Travel

Built for Mac OS X

Sig Software

NameCleaner

An industrial-strength utility to manipulate file names and types, focused on preparing Macintosh files for use under Windows and Unix, and vice versa. Available with previewing, logging, scheduling and scripting - download a trial version from our website for OS 7/8/9 or OS X.

www.sigsoftware.com
Email: sales@sigsoftware.com

InformINIT

The ultimate guide to the Classic Mac OS & the Classic Environment in Mac OS X

5 Mice/Stars–Macworld

Authoritative, a Must Have–MacAddict

Shareware of the Year–MacUser

The ultimate troubleshooting guide–Mac Today

The ultimate primer–ZDNet

www.InformINIT.com

Time Track

Keep track of every second of your time and bill for it with Time Track! Built in instructions make it easy to use. It is a simple way to manage your billable time for multiple projects and create a web page to show to your clients. Only $24.95 per single user license per platform. Finally! A versatile time tracking solution for Macintosh, Windows, and Palm!

www.trinfinitysoftware.com

Eudora Internet Mail Server (EIMS) 3.1 is the latest version of the most popular Internet mail server for the Macintosh. If you need to handle email for a dozen users, or thousands of users, EIMS is a reliable and easy to use solution.

EIMS 3.1 is available for US$400.00, there are no limits on the number of users that can be added, and free email support is included.

For more information, see
http://www.eudora.co.nz/

GraphicConverter converts pictures to different formats. Also it contains many useful features for picture manipulation.

See ***www.lemkesoft.com*** for more information.

Watson 1.5

Winner of the 2002 Apple Design Award for Most Innovative Mac OS X Product! An Aqua experience for the most useful Web services: TV & Movie Listings, Reference, Translation, Stocks, Flights, Package Tracking, and more. With an open architecture for third-party tools as well. Download the demo now!

www.karelia.com/watson

Trapcode - plug-ins for Adobe® After Effects®

Trapcode Shine is a fast light effect plug-in. The effect looks very much like volumetric light, but is actually a 2D effect. There are special controls to make shimmering lights and numerous coloring modes. This is an effect that you see everyday on TV and in many movie titles. Shine is available for Mac, Mac OSX and Windows.

www.trapcode.com

dragTHING

By TLA Systems

The Dock with more than one dimension.

Create multiple docks of any size, assign hot keys and even put the Trash back on the Desktop. A flexible and feature-laden tool for power-users. Runs natively on Mac OS X and 9 in five languages.

"...you made the switch to OS X a lot easier for me..." - Bob LeVitus

"...DragThing can rightfully be called an indispensable aid to working with your Mac..." - MacUser UK

Download a copy now from www.dragthing.com.

The Journal of Macintosh Technology & Development

MacTech

Got a great product sold through Kagi?

Promote your product through the very cost effective Kagi Showcase in MacTech Magazine.

For more information, contact us at

adsales@mactech.com

```
#include <Carbon/Carbon.h>

#include "jigsaw.h"

// make sure we aren't using printf in the non-console builds
#ifndef UI_CONSOLE
#define printf error
#endif

typedef struct PT PT; // a point
struct PT
{
  int x, y;
};
#define zNil 0x7FFF

typedef struct EDGE EDGE; // info about an edge
struct EDGE
{
  char fEdge;
};

typedef enum
{
  btMask,
  btEdge
} BT;

typedef struct ROT ROT;
struct ROT // info for a rotation of a piece
{
  int mruReject;
  EDGE aedge[4];
  BitMap bitmapMask;
  BitMap bitmapEdge;
};

typedef struct PCE PCE;
struct PCE // a single piece, including its four rotations
{
  long cpixel;
  ushort clr;
  char cEdgeSide;
  char irotUsed;
  ROT arot[4];
};

typedef struct PUZ PUZ;
struct PUZ // the puzzle solving state
{
  // challenge state maintained by host
  CS *pcs;

  // the image data, (first the input file, then the output file after extracting the pieces)
  BITS bits;
  int fChangedSize;

  // bitmap mask for the puzzle as we decompose then solve it
  BitMap bitmapMask;

  // the array of pieces
  int cpce;
  PCE *papce;

  // memory block from which we allocate the bitmaps of the individual pieces
  char *pabBuffer;
  long cbBufferAlloc;
  long cbBufferUsed;

  // the current solution state
  int ipceNext;
  Rect rectPrev;
  int xRightEdge, yBottomEdge;
  int cpceRow;
  int yTopUnsolved;
  long cpixelImage;
  long cpixelPuzzle;
  long cpixelSolved;
  AbsoluteTime ticksTotal;
  AbsoluteTime tickStart;
};

// define adjacency
typedef struct DIR DIR;
struct DIR // a direction to move to an adjacent pixel
{
  int dx, dy;
};

// directions to look when checking the left edge of a piece
static const DIR s_adirLeft[] =
{
  {  0,  1 },
  { -1,  1 },
  { -1,  0 },
  { -1, -1 },
  {  0, -1 },
};

// directions to look when checking the top edge of a piece
static const DIR s_adirAbove[] =
{
  {  1,  0 },
  {  1, -1 },
  {  0, -1 },
  { -1, -1 },
  { -1,  0 },
};

enum
{
  iedgeTop = 0,
  iedgeLeft,
  iedgeBottom,
  iedgeRight,
  cedge
};

// directions to look when finding the boundary of a piece
static const DIR s_adirEdge[cedge] =
{
  {  1,  0 },
  {  0,  1 },
  { -1,  0 },
  {  0, -1 },
};

// function prototypes deleted for brevity
```

SolveJigsaw

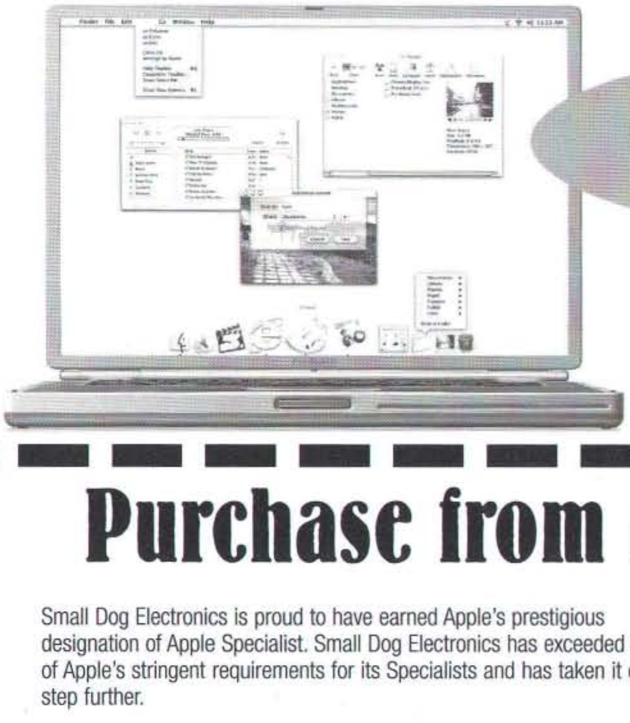
```
// the public entry point to the puzzle solver
void SolveJigsaw(CS *pcs, const char pszFile[])
{

  long cbRead;
  OSStatus ec;

  int cSolved = 0;
  FN fnOut = fnNil;
  double musecTotal = 0.0;

  // init the puzzle structure
  PUZ puz;
  memset(&puz, 0, sizeof(PUZ));
  puz.pcs = pcs;

  // open and read the challenge file (omitted for brevity)
  // create and open the output file (omitted for brevity)

  // process the cases
  for (; iCase <= cCase; ++iCase)
  {
    // note the starting time
    puz.ticksTotal = s_tickZero;
    puz.tickStart = UpTime();

#ifdef UI_CONSOLE
    printf("process case %d of %d:\n", iCase, cCase);
#endif

    // load in the image
    if (!_FLoadData(&puz, iCase))
      break;
```

Special Small Dogs

Here are just a few of the photos our customers have sent us of their special dog friends!

Check out all the other special Small Dog photos and send us one of your own at www.smalldog.com!

Small Dog's Special



PowerBook G4/667 Titanium
$3199

- 512MB RAM
- 30gb Hard Drive
- Combo Drive
- Airport card and base station

Purchase from an Apple Specialist!

Small Dog Electronics is proud to have earned Apple's prestigious designation of Apple Specialist. Small Dog Electronics has exceeded all of Apple's stringent requirements for its Specialists and has taken it one step further.

Small Dog sells only the most powerful personal computers in the world—every single one of them an Apple Macintosh! In addition to the rigorous Apple training program, each Small Dog employee has chosen a specific area of expertise to concentrate upon. You can be assured that whomever you talk to at Small Dog, from our sales engineers to our shipping department, you are talking to a trained, enthusiastic, Apple Macintosh Specialist!

Small Dog Electronics is also an Authorized Apple Service Provider Plus. We have certified Apple Technicians that utilize genuine Apple parts for all repairs and provide technical support for our customers based upon their experience in upgrading and supporting thousands of Apple Macintosh computers for our customers!

Talk To an Apple Professional!
Did you know that when you call Small Dog Electronics, you are most likely talking to an Apple Product Professional? Each year, we participate in Apple's on-line courses and seminar training programs to learn as much as we can about the products that we sell and service!

Exclusive Top Dog Membership Treats
For each dollar purchased on-line at the Small Dog web site, you will receive a doggie treat. You can redeem your accumulated treats for Small Dog or Apple apparel and other products. You are automatically enrolled and begin accumulating treats in the Top Dog Club with your first purchase. Remember the more you buy, the more treats for you!

Small Dog Electronics

1673 Main Street
Waitsfield, VT 05673 USA
Phone: 802-496-7171
Online: smalldog.com

Apple Specialist

```
#ifdef UI_GUI
      // put the starting image into the output window
      puz.ticksTotal = AddAbsoluteToAbsolute(puz.ticksTotal,
            SubAbsoluteFromAbsolute(UpTime(), puz.tickStart));
      DrawBits(puz.pcs, puz.bits, puz.xRightEdge,
puz.yBottomEdge);
      puz.tickStart = UpTime();
#endif

      // let's find all of the pieces
      if (!_FFindPieces(&puz))
      {
        _FreePuz(&puz);
        continue;
      }

      // place the first piece into the output buffer
      _PlacePce(&puz, 0, 0, 0, 0);
#ifdef UI_GUI
      puz.ticksTotal = AddAbsoluteToAbsolute(puz.ticksTotal,
            SubAbsoluteFromAbsolute(UpTime(), puz.tickStart));
      DrawBits(puz.pcs, puz.bits, puz.xRightEdge,
puz.yBottomEdge);
      puz.tickStart = UpTime();
#endif

      while (puz.ipceNext < puz.cpce && _FSolvePce(&puz))
      {
#ifdef UI_GUI
        puz.ticksTotal = AddAbsoluteToAbsolute(puz.ticksTotal,
            SubAbsoluteFromAbsolute(UpTime(), puz.tickStart));
        if (puz.fChangedSize)
        {
          DrawBits(puz.pcs, puz.bits, puz.xRightEdge,
            puz.yBottomEdge);
          puz.fChangedSize = fFalse;
        }
        else
        {
          UpdateBits(puz.pcs, puz.bits, puz.rectPrev);
        }
        puz.tickStart = UpTime();
#endif
      }

      // if we completely solved the puzzle, write out the solution
      if (puz.ipceNext == puz.cpce)
      {
#ifdef UI_CONSOLE
        printf("\tpuzzle %d solved\n", iCase);
#endif
        // write out our solution
        if (!_FWriteResult(&puz, iCase))
          break;

        ++cSolved;
      }

      if (puz.ipceNext < puz.cpce)
      {
#ifdef UI_CONSOLE
        printf("ERROR: Solve failed after %d pieces\n",
puz.ipceNext);
#endif
      }

      // free the memory used by this puzzle
      _FreePuz(&puz);

      // get the ending time and subtract the starting time to get elapsed time
      // write the time, in microseconds, to the file
      //     (omitted for brevity)
    }

#ifdef UI_CONSOLE
  printf("%d of %d puzzles solved, total time = %g\n",
cSolved, cCase, musecTotal);
#endif

LRet:
  _FreePuz(&puz); // harmless if the puz was freed through normal process
  if (fnOut != 0)
    CloseFn(fnOut);
}

// load in the data for a puzzle (omitted for brevity)
static int _FLoadData(PUZ *ppuz, int iCase)
//  (omitted for brevity)
```

_FFindPieces

```
// locate the pieces in the input file, build bitmaps for them
static int _FFindPieces(PUZ *ppuz)
{
  int fSuccess = fFalse;
  ppuz->cpixelPuzzle = 0;
  ppuz->yTopUnsolved = 0;

  ushort *psw;
  {
    const ushort *pswLim =
        &ppuz->bits.pasw[ppuz->bits.dx * ppuz->bits.dy];
    for (psw = ppuz->bits.pasw; psw < pswLim; ++psw)
      if (*psw > ppuz->cpce)
        ppuz->cpce = *psw;
  }

  // allocate and init the piece array
  ppuz->papce = new PCE[ppuz->cpce];
  if (ppuz->papce == NULL)
  {
#ifdef UI_CONSOLE
    printf("ERROR: oom allocating piece array with %d
entries\n",
          ppuz->cpce);
#endif
    goto LExit;
  }
  memset(ppuz->papce, 0, sizeof(PCE)*ppuz->cpce);

  psw = ppuz->bits.pasw;
  for (int y = 0; y < ppuz->bits.dy; ++y)
  {
    for (int x = 0; x < ppuz->bits.dx; )
    {
      int xStart = x++;
      ushort clr = *psw++;
      if (clr == 0)
        continue;
      while (x < ppuz->bits.dx && *psw == clr)
        ++x, ++psw;

      PCE *ppce = &ppuz->papce[clr-1];
      if (ppce->clr == 0)
      {
        ppce->clr = clr;
        _SetRect(&ppce->arot[0].bitmapMask.bounds, xStart,
            y, x, y+1);
      }
      else
      {
        _ExpandRect(&ppce->arot[0].bitmapMask.bounds,
            y, xStart, x);
      }
    }
  }

  // figure out an upper bound on how much bitmap data we'll need
  {
    int dzMost = 0;
    for (int ipce = 0; ipce < ppuz->cpce; ++ipce)
    {
      Rect *prect = &ppuz-
>papce[ipce].arot[0].bitmapMask.bounds;
      int dx = prect->right - prect->left;
      if (dx > dzMost)
        dzMost = dx;
      int dy = prect->bottom - prect->top;
      if (dy > dzMost)
        dzMost = dy;
    }
    long cbBitmapMost = dzMost * ((dzMost + 15) >> 4) << 1;
    ppuz->cbBufferAlloc = cbBitmapMost * ppuz->cpce * 8;
    ppuz->pabBuffer = new char[ppuz->cbBufferAlloc];
  }
```

```
   for (int ipce = 0; ipce < ppuz->cpce; ++ipce)
   {
      if (!_FBuildPceBitmaps(ppuz, ipce))
         goto LExit;
#ifdef UI_GUI
      ppuz->ticksTotal = AddAbsoluteToAbsolute(ppuz-
>ticksTotal,
            SubAbsoluteFromAbsolute(UpTime(), ppuz->tickStart));
      UpdateBits(ppuz->pcs, ppuz->bits, ppuz-
>papce[ipce].arot[0].bitmapMask.bounds);
      ppuz->tickStart = UpTime();
#endif
   }

   fSuccess = fTrue;

#ifdef UI_CONSOLE
   printf("\tfound %d pieces comprising %ld pixels\n", ppuz-
>cpce, ppuz->cpixelPuzzle);
#endif

LExit:
   return fSuccess;
}
```

_FBuildPceBitmap

```
// build the required bitmaps for a puzzle piece in all four rotations
static int _FBuildPceBitmaps(PUZ *ppuz, int ipce)
{
   int fSuccess = fFalse;
   PCE *ppce = &ppuz->papce[ipce];

   Rect rectBounds = ppce->arot[0].bitmapMask.bounds;

   if (!_FCreateBitmap(ppuz, &ppce->arot[0].bitmapMask,
            rectBounds.left, rectBounds.top, rectBounds.right,
            rectBounds.bottom))
      goto LExit;
   if (!_FCreateBitmap(ppuz, &ppce->arot[0].bitmapEdge,
            rectBounds.left, rectBounds.top, rectBounds.right,
            rectBounds.bottom))
      goto LExit;

   ppuz->cpixelPuzzle += ppce->cpixel = _ColorFill(&ppuz-
>bits,
            rectBounds, ppce->clr, &ppce->arot[0].bitmapMask);
   _BuildEdgeBitmap(ppce->arot[0].bitmapMask,
            &ppce->arot[0].bitmapEdge);

   _GetEdgeInfo(ppuz, ppce);

   for (int irot = 1; irot < DIM(ppce->arot); ++irot)
      if (!_FRotateBitmap(ppuz, ppce->arot[0].bitmapEdge, irot,
            &ppce->arot[irot].bitmapEdge))
         goto LExit;

   fSuccess = fTrue;

LExit:
   return fTrue;
}
```

_FSolvePce

```
// solve for one piece of the puzzle
static int _FSolvePce(PUZ *ppuz)
{
   // find the first unplaced pixel
   int xTarget, yTarget;

   // start x out just to the right of the most recent piece we placed,
   // unless that piece hit the right edge of the puzzle
   xTarget = ppuz->rectPrev.right;
   if (xTarget == ppuz->xRightEdge)
      xTarget = 0;
   else
      ++xTarget;

   // starting at the top of the unsolved area of the puzzle, march down
   // the chosen x column until we find an unset pixel
   for (yTarget = ppuz->yTopUnsolved; yTarget < ppuz-
>yBottomEdge;
```

Turbocharge development with RADicode

RADicode is a RAD tool that converts REALbasic project files into functional PowerPlant GUIs using PowerPlant best coding practices and conventions. Get your application from drawing board to working code faster than ever before. Compile and run the generated project, and go home early.

Introductory Price

$149.95*

SRP $299.99

solidwave
(617) 776-8155
www.solidwave.com

*Offer ends August 31, 2002

```
          ++yTarget)
      if (!_FGetBit(ppuz->bitmapMask, xTarget, yTarget))
        break;

    // now, try to move as far up and to the left as possible since
    // we'd really like to have the upper left corner pixel of the next
    // piece as our target
    while (fTrue)
    {
      if (xTarget > 0 && !_FGetBit(ppuz->bitmapMask, xTarget-
1,
          yTarget))
        --xTarget;
      else if (yTarget > 0 && !_FGetBit(ppuz->bitmapMask,
xTarget,
          yTarget-1))
        --yTarget;
      else
        break;
    }

    // finally, make sure the target point in the leftmost unset pixel in yTarget's row
    // so that we only have to check the leftmost set pixel of each scan line in each
    // candidate piece
    for (int xT = xTarget - 1; xT >= ppuz->rectPrev.left; --
xT)
    {
      if (!_FGetBit(ppuz->bitmapMask, xT, yTarget))
        xTarget = xT;
    }

    PT aptCheck[4];
    int cptCheck = 0;

    if (ppuz->rectPrev.right < ppuz->xRightEdge)
    {
      PCE *ppcePrev = &ppuz->papce[ppuz->ipceNext-1];
      ROT *prot = &ppcePrev->arot[ppcePrev->irotUsed];
      int cptIn = _CptGetEdgeShape(prot->bitmapMask, aptCheck);
      for (int ipt = 0; ipt < cptIn; ++ipt)
      {
        PT pt = aptCheck[ipt];
        _OffsetPt(&pt,
          ppuz->rectPrev.left - prot->bitmapEdge.bounds.left,
          ppuz->rectPrev.top  - prot->bitmapEdge.bounds.top
          );
        pt.x += 1;
        if (_FPtInRect(ppuz->bitmapMask.bounds, pt.x, pt.y) &&
              !_FGetBit(ppuz->bitmapMask, pt.x, pt.y))
          aptCheck[cptCheck++] = pt;
      }
    }
    if (ppuz->yTopUnsolved > 0)
    {
      const PCE *ppceAbove =
          &ppuz->papce[ppuz->ipceNext - ppuz->cpceRow];
      const ROT *protAbove = &ppceAbove->arot[ppceAbove-
>irotUsed];
      PT ptTop;
      ptTop.x = (protAbove->bitmapMask.bounds.left +
          protAbove->bitmapMask.bounds.right)/2;
      for (ptTop.y = ppuz->yTopUnsolved;
              ptTop.y < ppuz->yBottomEdge; ++ptTop.y)
      {
        if (!_FGetBit(ppuz->bitmapMask, ptTop.x, ptTop.y))
        {
          aptCheck[cptCheck++] = ptTop;
          break;
        }
      }
    }

    for (int pass = 1; pass <= 2; ++pass)
    {
      for (int irot = 0; irot < DIM(ppuz->papce[0].arot);
++irot)
      {
        for (int ipce = ppuz->ipceNext; ipce < ppuz->cpce;
++ipce)
        {
          PCE *ppce = &ppuz->papce[ipce];
          ROT *prot = &ppce->arot[irot];

          // if we're looking for an edge piece, consider only edge pieces
          if (yTarget == 0 || xTarget == 0)
          {
            if (ppce->cEdgeSide == 0)
              continue;
            if (yTarget == 0 && !prot->aedge[iedgeTop].fEdge)
              continue;
            if (xTarget == 0 && !prot->aedge[iedgeLeft].fEdge)
              continue;
          }

          if (pass == 2 && prot->mruReject != ppuz->ipceNext)
          {
            // some pieces are too wierd to work with the innie/outie testing
(below),
            // let all the configurations that got rejected previously run through
            // this time, but don't test any piece twice!
            continue;
          }

          Rect bounds = prot->bitmapEdge.bounds;
          for (int y = WMin(bounds.bottom-1,
            bounds.top + (yTarget - ppuz->yTopUnsolved));
            y >= bounds.top; --y)
          {
            for (int x = bounds.left; x < bounds.right; ++x)
            {
              if (!_FGetBit(prot->bitmapEdge, x, y))
                continue;

              int dx = xTarget - x;
              int dy = yTarget - y;

              // make sure the proposed bounding box fits inside the puzzle
              if (bounds.left + dx < 0
                || bounds.right + dx > ppuz->xRightEdge
                || bounds.top + dy < 0
                || bounds.bottom + dy > ppuz->yBottomEdge)
              {
                continue;
              }

              if (pass == 1)
              {
                // make sure we have edge bits to join with check points on the
                // previous piece
                int ipt;
                for (ipt = 0; ipt < cptCheck; ++ipt)
                  if (!_FGetBitSafe(prot->bitmapEdge,
                      aptCheck[ipt].x - dx, aptCheck[ipt].y -
dy))
                    break;
                if (ipt < cptCheck)
                {
                  prot->mruReject = ppuz->ipceNext;
                  break;
                }
              }

              if (_FTestPce(ppuz, ipce, irot, dx, dy))
              {
                _PlacePce(ppuz, ipce, irot, dx, dy);
                              return fTrue;
              }
              break;
            }
          }
        }
      }
    }

    return fFalse;
}
```

_FTestPce

```
// test whether or not the indicated piece fits into the puzzle at
// the specified offset from it starting position.
static int _FTestPce(PUZ *ppuz, int ipce, int irot,
            int dx, int dy)
{
  int fPassed = fFalse;
```

Take

C o N t R o L

With JobOrder

Estimating ▪ Scheduling ▪ Job Costing ▪ Traffic ▪ Billing
Reporting ▪ Notifications ▪ Accounting ▪ Calendars

[Discover the Affordable Management Software]

free demo at www.JobOrder.com
toll-free 877.714.2587 phone 607.756.4150

JobOrder™
maximizing business productivity

```
  // check that no edge pixel invades the completed portion of the puzzle
  PCE *ppce = &ppuz->papce[ipce];
  ROT *prot = &ppce->arot[irot];
  BitMap *pbitmapEdge = &prot->bitmapEdge;
  Rect bounds = pbitmapEdge->bounds;

  for (int y = bounds.top; y < bounds.bottom; ++y)
  {
    uchar mask;
    int x = bounds.left;
    uchar *pb = _PbMaskForBitmapBit(pbitmapEdge, x, y,
&mask);
    for ( ; x < bounds.right; )
    {
      char b = *pb++;
      if (b == 0)
      {
        x += 8;
        continue;
      }
      for ( ; mask != 0; ++x, mask >>= 1)
      {
        if (b & mask)
        {
          if (!_FPtInRect(ppuz->bitmapMask.bounds, x+dx,
y+dy) ||
                  _FGetBit(ppuz->bitmapMask, x+dx, y+dy))
          {
            goto LRet;
          }
        }
      }
      mask = 0x80;
    }
  }

  // make sure the mask bitmap has been built
  BitMap *pbitmapMask;
  pbitmapMask = &prot->bitmapMask;
  if (pbitmapMask->baseAddr == NULL)
  {
    if (!_FRotateBitmap(ppuz, ppce->arot[0].bitmapMask, irot,
            pbitmapMask))
    {
#ifdef UI_CONSOLE
      printf("ERROR: oom building rotated mask bitmap for
piece %d\n", ppuz->ipceNext);
#endif
      goto LRet;
    }
  }

  // now scan left edge pixels looking for pixels that are interior to the proposed
  // assembled pieces in order to pass, there have to be interior edge pixels from the
  // top of the piece down "most" the piece, with no gaps.
  {
    int xMid = (bounds.left + bounds.right)/2;
    int yInteriorLim = bounds.top;
    int yInteriorReqd = (bounds.top + bounds.bottom)/2;
    for (int y = bounds.top; y < bounds.bottom; ++y)
    {
      int x = bounds.left;
      uchar mask;
      uchar *pb = _PbMaskForBitmapBit(pbitmapEdge,
bounds.left,
                              y, &mask);
      for ( ; x < xMid; )
      {
        char b = *pb++;
        if (b == 0x00)
        {
          x += 8;
          continue;
        }
        for ( ; mask != 0; ++x, mask >>= 1)
        {
          if (b & mask)
          {
            if (x >= xMid)
              break;
            int idir;
            for (idir = 0; idir < DIM(s_adirLeft); ++idir)
            {
              struct DIR dir = s_adirLeft[idir];
              if (_FPtInRect(ppuz->bitmapMask.bounds,
x+dx+dir.dx,
                      y+dy+dir.dy)
                && !_FGetBit(ppuz->bitmapMask, x+dx+dir.dx,
                      y+dy+dir.dy)
                && !_FGetBitSafe(*pbitmapMask, x+dir.dx,
y+dir.dy)
                )
              {
                break;
              }
            }
            if (idir == DIM(s_adirLeft))
            {
              // this pixel is an interior pixel
              if (yInteriorLim < y)
              {
                goto LRet; // we lose, there was a gap in the interior
                           //edge pixels
              }
              yInteriorLim = y+1;
              x = bounds.right; // force the "x" loop to terminate
              break;
            }
            else
            {
              // this pixel is not an interior pixel
              if (y < yInteriorReqd)
                goto LRet; // we didn't find enough contiguous interior
                           //edge pixels

              // we don't mind that this edge pixel isn't an interior pixel,
              // and we can stop looking at this scan line
              goto LPassedLeft;
            }
          }
        }
        mask = 0x80;
      }

      Assert (x >= xMid);
      // any pixels on the left half of this scan line are interior pixels
      if (yInteriorLim < y)
        goto LRet; // we lose, there was a gap in the interior edge pixels
      yInteriorLim = y+1;
    }
  }
LPassedLeft:

  // now scan top edge pixels looking for pixels that are interior to the proposed
  // assembled pieces in order to pass, there have to be interior edge pixels from the
  // top of the piece down "most" of the piece, with no gaps.
  {
    int yMid = (bounds.top + bounds.bottom)/2;
    int xInteriorLim = bounds.left;
    int xInteriorReqd = (bounds.left + bounds.right)/2;
    for (int x = bounds.left; x < bounds.right; ++x)
    {
      int y = bounds.top;
      uchar mask;
      uchar *pb = _PbMaskForBitmapBit(pbitmapEdge, x, y,
&mask);

      for ( ; y < yMid; ++y, pb += pbitmapEdge->rowBytes)
      {
        if (*pb & mask)
        {
          int idir;
          for (idir = 0; idir < DIM(s_adirAbove); ++idir)
          {
            struct DIR dir = s_adirAbove[idir];
            if (_FPtInRect(ppuz->bitmapMask.bounds,
x+dx+dir.dx,
                        y+dy+dir.dy)
              && !_FGetBit(ppuz->bitmapMask, x+dx+dir.dx,
                        y+dy+dir.dy)
              && !_FGetBitSafe(*pbitmapMask, x+dir.dx,
y+dir.dy)
              )
```

```
            {
              break;
            }
          }
          if (idir == DIM(s_adirAbove))
          {
            // this pixel is an interior pixel
            if (xInteriorLim < x)
              goto LRet; // we lose, there was a gap in the interior edge
                        //pixels

            xInteriorLim = x+1;
            y = bounds.bottom; // force the "y" loop to terminate
            break;
          }
          else
          {
            // this pixel is not an interior pixel
            if (x < xInteriorReqd)
              goto LRet; // we didn't find enough contiguous interior
                        //edge pixels

            // we don't mind that this edge pixel isn't an interior pixel,
            // and we can stop looking at this scan line
            goto LPassedTop;
          }
        }
      }

      Assert (y >= yMid);
      // any pixels on the top half of this scan line are interior pixels
      if (xInteriorLim < x)
        goto LRet; // we lose, there was a gap in the interior edge pixels
      xInteriorLim = x+1;
    }
  }

LPassedTop:
  fPassed = fTrue;

LRet:
  return fPassed;
}

// write out the solved puzzle state
static int _FWriteResult(PUZ *ppuz, int iCase)
// omitted for brevity
```

_PlacePce

```
// place the indicated piece into the puzzle at the specified offset
// from its current position
static void _PlacePce(PUZ *ppuz, int ipce, int irot, int
dxOffset, int dyOffset)
{
  PCE *ppce = &ppuz->papce[ipce];
  ROT *prot = &ppce->arot[irot];
  BitMap *pbitmapMask = &prot->bitmapMask;
  Rect boundsSrc = pbitmapMask->bounds;
  Rect boundsDst = boundsSrc;
  _OffsetRect(&boundsDst, dxOffset, dyOffset);

  // record the bounds of where we placed the most current piece
  ppuz->rectPrev = boundsDst;
  ppce->irotUsed = irot;

  // copy the piece to the output image
  _TransferBitmapToImage(ppuz, prot, btMask, ppce->clr,
          dxOffset, dyOffset);

  // check to see if we've found the right edge yet
  if (ppuz->xRightEdge == ppuz->bits.dx &&
          prot->aedge[iedgeRight].fEdge)
  {
    // test to see if the piece we just put in is an edge piece,
    // we require a more strict "edge" test for this purpose
    int y;
    int x = boundsSrc.right - 1;
    for (y = boundsSrc.top; y < boundsSrc.bottom; ++y)
      if (!_FGetBit(*pbitmapMask, x, y))
        break;
    if (y > (boundsSrc.top + 2*boundsSrc.bottom)/3)
    {
```

Made with

REALbasic®

REAL Software and MacTech present the REALbasic Showcase to highlight some of the fantastic solutions created by REALbasic users worldwide.The showcase illustrates the wide range of applications that developers using REALbasic can create. Some benefit any Mac user, and others are more specific.All of them are seriously cool!

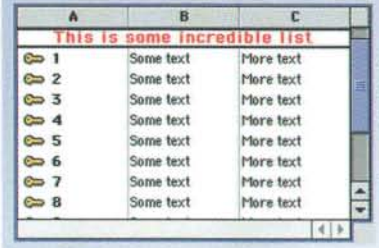
REALbasic is the powerful, easy-to-use tool for creating your own software for Macintosh, Mac OS X, and Windows. It runs natively on Mac OS X as well as earlier versions of the Mac OS. For more information, please visit: **<www.realbasic.com>**.

The Made with REALbasic program is a cooperative effort between REALbasic users and REAL Software, Inc. to promote the products created using REALbasic and the people who create them. For more information about the Made with REALbasic program, please visit: **<www.realbasic.com/realbasic/mwrb/Partners/MwRbProgram.html>**.

Extend REALbasic with Advanced Plugins

Spreadsheet Controls:

We provide a wide range of spreadsheet controls for REALbasic, including multistyled and custom rendering spreadsheet controls.



	A	B	C
	This is some incredible list		
1		Some text	More text
2		Some text	More text
3		Some text	More text
4		Some text	More text
5		Some text	More text
6		Some text	More text
7		Some text	More text
8		Some text	More text

- More than 32k of rows.
- Classic, OS X and Win32.
- Accelerated for maximum speed.
- Images in cells.

Cryptography:

We provide data encryption, encoding, compression and hashing for REALbasic.

Supported Algorithms:

- Encryption:
  - e-CryptIt
  - BlowFish (448 bit)
  - AES (Rijndael) (256 bit)
- Encoding:
  - e-CryptIt Flexible
  - Base 64
  - BinHex
  - MacBinary III
  - AppleSingle / Double
  - UUCoding
- Compression:
  - Zip on Strings and streams (.gz)
- Hashing and Checksums:
  - CRC32 / Adler32
  - MD5 / HMAC_MD5
  - SHA / SHA1 / HMAC_SHA1

Other Plugins:

We have many other plugins for REALbasic, including plugins to do advanced MacOS Toolbox tasks and more custom Controls.

Speed up developement and make more advanced applications by using plugins ! Get free demos at www.einhugur.com

Einhugur Software
sales@einhugur.com
www.einhugur.com

db Reports

Printing database driven reports from your REALbasic project has never been easier.

Powerful expressions with an easy-to-use interface make this an indispensable tool for the REALbasic programmer providing business solutions.

Download a free demo today.

http://abDataTools.com

Whistle Blower

Enterprise server monitor and restart utility
whistleblower.sentman.com

Connect to and validate the response from web servers, cgi scripts and over 23 other types of servers.

Send email, pages and perform unattended restarts via MasterSwitch or PowerKey.

Shifts make sure that the person on call when the server goes down is the one who gets the page.

68k, PPC and Carbon

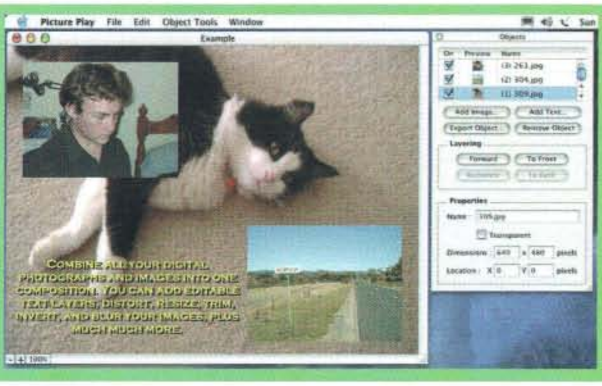
Web based administration lets you check on and restart your servers from anywhere.

Customize your response to an outage with Apple Script.

email us at whistleblower@sentman.com

Picture Play

Create digital image compositions on your Mac, quickly and easily.


Picture Play
File
Edit
Object Tools
Window
Example
Objects
Add Image...
Add Text...
Export Object...
Remove Object
Layering
Forward
To Front
Properties
Name
Transparent
Dimensions
Location
Combine all your digital photographs and images into one composition. You can add editable text layers, distort, resize, trim, invert, and blur your images, plus much much more.

For more information, email chris@crescendosw.com, or
visit us today at www.crescendosw.com!

the cross-platform solution
for Macintosh and Windows

Version 3.0 featuring OS X coming soon !
Mac designers : build professional screensavers without touching a Windows machine !

• build both Macintosh and Windows screensavers with a single click, no matter what system* you are using !

• use any QuickTime 6.0 movie format :
Macromedia Flash 5.0, MPEG, Cinepak, MP3, Midi, AVI, DV Video...
or, now with version 3.0, build your own basic Slide Shows !

• include a hidden movie that can be unlocked with a registration code

• customize and fully-brand both Installers and Screensaver control panels with pictures and text

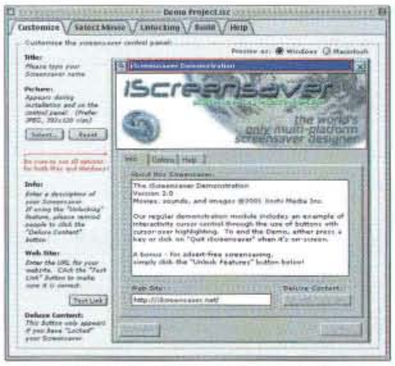
• Screensavers install without DLLs, extensions, or restarts

simple WYSIWYG editor

supports interactive Flash and QuickTime

consistent cross-platform user interface

try before you buy fully functional online downloads



the iScreensaver Designer editing environment

creating screensavers
for both Windows and Macintosh
has <u>never</u> been this easy

http://iScreensaver.net
email : info @ iScreensaver.net

* supported systems, as of June 2002, include:
Macintosh OS 8.6 to 9.2.2, Microsoft Windows 95/98/ME, NT4/NT2000/XP
iScreensaver Designer version 3.0 will support up to Macintosh OS 10.2

Corona

accounting software

"Easy to set up, easy to use, and excellent support from the developer."

Five stars on VersionTracker.com
Four cows on Tucows.com

• cash entry
• invoicing
• payroll
• reports

$64.95

Version 1.9 now with: sales tax accounting
"keyless entry" invoices
drag 'n drop accounts

Free 30-day trial
http://homepage.mac.com/idlewild/CoronaUS.hqx

A best friend for business!

p.o. box 472 • aurora • oregon • 97002
idlewild@mac.com

LIGHTHOUSE The Time-Saving Internet Assistant

Spending long hours online searching for code snippets and developer docs? Get organized with Lighthouse.

	STANDARD EDITION FREE	PLATINUM EDITION $10 US
Search the Web with free plugins or create your own search plugins	✓	✓
Store Account Info for access to membership sites & web mail	✓	✓
Plugins Manager to add / edit Lighthouse plugins	✓	✓
Web Favorites Organizer offers advanced bookmarking features		✓
Note Keeper saves code snippets, ideas, research notes, etc.		✓
Local Weather at your fingertips for home or travel		✓
Calendar & World Clock Personalized reference tools		✓

FREE DOWNLOAD
for Classic Mac, Mac OS X and Microsoft Windows

www.ebutterfly.com

electric butterfly

TitleTrack™ JUKEBOX

MAC MUSIC MANAGEMENT FOR SONY® 5 TO 400 DISC CD CHANGERS

Control up to 12 changers (4,800 music CDs)
Control Any Brand Stereo Receiver
Play MP3 and other Sound Files
Plus much, much more!!!

www.titletrack.com **info@titletrack.com**

The Journal of Macintosh Technology & Development

MacTech®

Are you a REALBasic Developer?

Promote your product through the very cost effective REALBasic Showcase in MacTech Magazine.

For more information, contact us at

adsales@mactech.com

piPop

Fast, Simple, Efficient

Manage - Navigate - Launch

Custom Hierarchical
Pop-up Menus for your Mac
(OS 8.6 and Higher)

piDog Software
http://www.pidog.com/

```
      for ( ; y < boundsSrc.bottom; ++y)
        if (_FGetBit(*pbitmapMask, x, y))
          break;
      if (y == boundsSrc.bottom)
      {
        if ((ppuz->cpixelPuzzle % boundsDst.right) == 0
          && (ppuz->cpce % (ppuz->ipceNext+1)) == 0)
        {
          ppuz->xRightEdge = boundsDst.right;
          ppuz->yBottomEdge = ppuz->cpixelPuzzle/ppuz-
>xRightEdge;
          ppuz->cpceRow = ppuz->ipceNext+1;
          ppuz->fChangedSize = fTrue;
        }
      }
    }
  }

  // if we hit the right edge or hit a pixel in the first unsolved scan line,
  // update ppuz->yTopUnsolved
  if (ppuz->rectPrev.right == ppuz->xRightEdge ||
          boundsSrc.top+dyOffset == ppuz->yTopUnsolved)
  {
    for ( ; ; ++ppuz->yTopUnsolved)
    {
      uchar mask;
      const uchar *pb = _PbMaskForBitmapBit(&ppuz-
>bitmapMask,
          0, ppuz->yTopUnsolved, &mask);
      int x;
      for (x = 0; x < ppuz->xRightEdge; x += 8, ++pb)
      {
        if (*pb == 0xFF)
          continue;
        for ( ; mask != 0; mask >>= 1, ++x)
          if ((*pb & mask) == 0)
            break;
        if (mask != 0)
          break;
      }
      if (x < ppuz->xRightEdge)
        break;
    }
  }

  // update stats
  ppuz->cpixelSolved += ppce->cpixel;

  // move the rotation we used to its correct location for use later
  _OffsetRect(&prot->bitmapMask.bounds, dxOffset, dyOffset);
  _OffsetRect(&prot->bitmapEdge.bounds, dxOffset, dyOffset);
```

```
  // update the piece array
  if (ipce != ppuz->ipceNext)
  {
    PCE pceT = ppuz->papce[ipce];
    ppuz->papce[ipce] = ppuz->papce[ppuz->ipceNext];
    ppuz->papce[ppuz->ipceNext] = pceT;
  }
  ++ppuz->ipceNext;

}

// set the mask bitmap and color in the solved puzzle with the new piece
static void _TransferBitmapToImage(PUZ *ppuz, const ROT
*prot,
    BT bt, ushort clr, int dxOffset, int dyOffset)
// omitted for brevity
```

_FreePuz

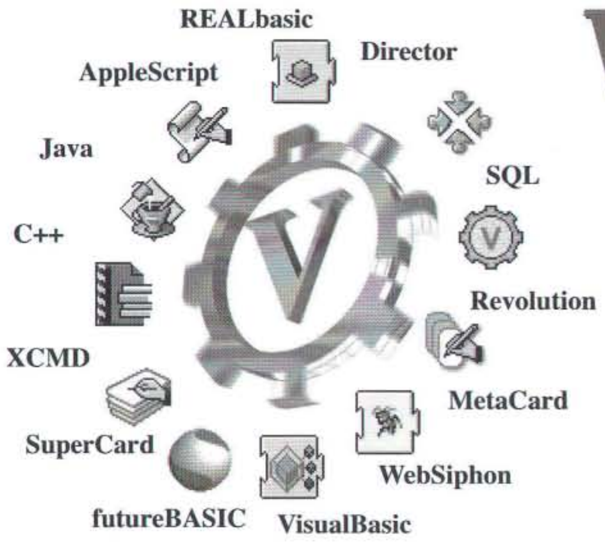
```
// free the puzzle state and everything it allocated
static void _FreePuz(PUZ *ppuz)
{
  _FreeBits(&ppuz->bits);
  _FreeBitmap(&ppuz->bitmapMask);

  if (ppuz->papce != NULL)
  {
    delete [] ppuz->pabBuffer;
    ppuz->pabBuffer = NULL;
    ppuz->cbBufferUsed = ppuz->cbBufferAlloc = 0;

    delete [] ppuz->papce;
    ppuz->papce = NULL;
    ppuz->cpce = ppuz->ipceNext = 0;
  }

}
```

_FreeBits

```
// free the buffer that holds the bitmap image data
static void _FreeBits(BITS *pbits)
{
  if (pbits->pasw != NULL)
  {
    delete [] pbits->pasw;
    pbits->pasw = NULL;
  }
}
```

_FCreateBitmap

```
// allocate a new bitmap with the specified bounding rect
// if ppuz != NULL, use the preallocted buffer, otherwise use new
static int _FCreateBitmap(PUZ *ppuz, BitMap *pbitmap, int
```



Valentina

Object-Relational SQL Database

The fastest database engine for MacOS/Windows

It operates 100's and sometimes a 1000 times faster than other systems

www.paradigmasoft.com

Hosted by macserve.net
Download full featured evaluation version

```
xLeft, int yTop, int xRight, int yBottom)
{
  _SetRect(&pbitmap->bounds, xLeft, yTop, xRight, yBottom);
  pbitmap->rowBytes = (((xRight - xLeft) + 15) >> 4) << 1;
  long cbBitmap = _CbBitmap(*pbitmap);

  if (ppuz != NULL)
  {
    // grab the chunk we need
    pbitmap->baseAddr = &ppuz->pabBuffer[ppuz->cbBufferUsed];
    ppuz->cbBufferUsed += cbBitmap;
  }
  else
  {
    pbitmap->baseAddr = new char[cbBitmap];
  }
  memset(pbitmap->baseAddr, 0, cbBitmap);

  return pbitmap->baseAddr != NULL;
}
```

_FreeBitmap

```
// free a bitmap whose bits were individually allocated
// be careful: most bit maps get allocated from the preallocated buffer
static void _FreeBitmap(BitMap *pbitmap)
{
  if (pbitmap->baseAddr != NULL)
  {
    delete [] pbitmap->baseAddr;
    pbitmap->baseAddr = NULL;
  }
}
```

_FGetBit

```
// return the state of the specified bit
static int _FGetBit(const BitMap &bitmap, int x, int y)
{
  uchar mask;
  uchar *pb = _PbMaskForBitmapBit(&bitmap, x, y, &mask);

  int fSet = ((*pb) & mask) != 0;
  return fSet;
}
```

_FGetBitSafe

```
// return the value of the specified bit if it's withing the bitmap,
// otherwise return 0
static int _FGetBitSafe(const BitMap &bitmap, int x, int y)
{
  int fSet = _FPtInRect(bitmap.bounds, x, y) ?
                _FGetBit(bitmap, x, y) : 0;
  return fSet;
}
```

_SetBit

```
// makes sure the indicated bit is set, returns non-zero if the bit needed to be set
static void _SetBit(BitMap *pbitmap, int x, int y)
{
  uchar mask;
  uchar *pb = _PbMaskForBitmapBit(pbitmap, x, y, &mask);

  *pb |= mask;
}

// set a bit in the specified bitmap
static void _SetBit(BitMap *pbitmap, int x, int y, int fSet)
{
  uchar mask;
  uchar *pb = _PbMaskForBitmapBit(pbitmap, x, y, &mask);

  if (fSet)
    *pb |= mask;
  else
    *pb &= ~mask;
}
```

_CbBitmap

```
// return size needed for the specified bitmap's bitmap data
static long _CbBitmap(const BitMap &bitmap)
{
  return (bitmap.bounds.bottom - bitmap.bounds.top) *
bitmap.rowBytes;
}
```

_PbMaskForBitmapBit

```
// for the specified pixel location within the bitmap, return the pointer
// to the correct byte in the bitmap data and the mask for the specified bit
// note - this is done incrementally in time critical routines
static uchar *_PbMaskForBitmapBit(const BitMap *pbitmap, int
x, int y, uchar *pmask)
{
  y -= pbitmap->bounds.top;
  x -= pbitmap->bounds.left;
  *pmask = 0x80 >> (x & 0x7);
  return (uchar *)&pbitmap->baseAddr[y * pbitmap->rowBytes +

(x>>3)];
}
```

_OffsetPt

```
// offset the point by the specified amounts
static void _OffsetPt(PT *ppt, int dx, int dy)
{
  ppt->x += dx;
  ppt->y += dy;
}
```

_FPtInRect

```
// is the specified point in the specified rect?
static int _FPtInRect(const Rect &rect, int x, int y)
{
  return rect.left <= x && x < rect.right &&
                  rect.top <= y && y < rect.bottom;
}
```

_SetRect

```
// like QD routine, but thread safe
static void _SetRect(Rect *prect, short left, short top,
short right, short bottom)
{
  prect->left = left;
  prect->top = top;
  prect->right = right;
  prect->bottom = bottom;
}
```

_OffsetRect

```
// like QD routine, but thread safe
static void _OffsetRect(Rect *prect, short dx, short dy)
{
  prect->left += dx;
  prect->right += dx;
  prect->top += dy;
  prect->bottom += dy;
}
```

_ExpandRect

```
// expand the specified rect to include the specified scan line
static void _ExpandRect(Rect *prect, int y, int xFirst, int
xLim)
{
  prect->bottom = y + 1;
  if (prect->left > xFirst)
    prect->left = xFirst;
  if (prect->right < xLim)
    prect->right = xLim;
}

// find all of the pixels in the original puzzle image data which match the
// specified "color", constraining the search to the specified rectangle
static long _ColorFill(BITS *pbits, const Rect &rectBounds,
ushort clrMatch, BitMap *pbitmapMask)
{
  ushort *psw = &pbits->pasw[rectBounds.top * pbits->dx +
          rectBounds.left];
  int cswSkip = pbits->dx - (rectBounds.right -
rectBounds.left);
  long cpixelMatch = 0;
  for (int y = rectBounds.top; y < rectBounds.bottom; ++y)
  {
    for (int x = rectBounds.left; x < rectBounds.right;
          ++x, ++psw)
    {
      if (*psw == clrMatch)
      {
        *psw = 0;
        _SetBit(pbitmapMask, x, y);
        ++cpixelMatch;
      }
    }
    psw += cswSkip;
  }

  return cpixelMatch;
}
```

_BuildEdgeBitmap

// find the boundary of a piece and construct a bitmap with those bits set

```
static void _BuildEdgeBitmap(const BitMap &bitmapMask, BitMap
*pbitmapEdge)
{
  for (int y = bitmapMask.bounds.top;
          y < bitmapMask.bounds.bottom; ++y)
  {
    for (int x = bitmapMask.bounds.left;
          x < bitmapMask.bounds.right; ++x)
    {
      if (_FGetBit(bitmapMask, x, y))
      {
        for (int idir = 0; idir < DIM(s_adirEdge); ++idir)
        {
          const struct DIR *pdir = &s_adirEdge[idir];
          if (!_FGetBitSafe(bitmapMask, x+pdir->dx, y+pdir-
>dy))
          {
            _SetBit(pbitmapEdge, x, y);
            break;
          }
        }
      }
    }
  }
}
```

_FRotateBitmap

// rotate the given bitmap by the specified number of quarter turns and put
// the result in a newly allocated bitmap

```
static int _FRotateBitmap(PUZ *ppuz, const BitMap &bitmapSrc,
        int irot, BitMap *pbitmapDst)
{
  Rect rectBounds = bitmapSrc.bounds;

  switch (irot)
  {
  case 1:
    if (!_FCreateBitmap(ppuz, pbitmapDst, rectBounds.top,
            rectBounds.left, rectBounds.bottom,
rectBounds.right))
        return fFalse;
      _RotateBitmap90(bitmapSrc, pbitmapDst);
      break;
  case 2:
    if (!_FCreateBitmap(ppuz, pbitmapDst, rectBounds.left,
            rectBounds.top, rectBounds.right,
rectBounds.bottom))
        return fFalse;
      _RotateBitmap180(bitmapSrc, pbitmapDst);
      break;
  case 3:
    if (!_FCreateBitmap(ppuz, pbitmapDst, rectBounds.top,
            rectBounds.left, rectBounds.bottom,
rectBounds.right))
        return fFalse;
      _RotateBitmap270(bitmapSrc, pbitmapDst);
      break;
  }

  return fTrue;
}
```

_RotateBitmap180

// rotate the source bitmap by 180 degrees into the destination bitmap

```
static void _RotateBitmap180(const BitMap &bitmapSrc,
          BitMap *pbitmapDst)
{
  for (int ySrc = bitmapSrc.bounds.top,
                          yDst = pbitmapDst-
>bounds.bottom-1;
                      ySrc < bitmapSrc.bounds.bottom;
++ySrc, --yDst)
    for (int xSrc = bitmapSrc.bounds.left,
                          xDst = pbitmapDst->bounds.right-1;
                      xSrc < bitmapSrc.bounds.right;
    ++xSrc, --xDst)
      _SetBit(pbitmapDst, xDst, yDst, _FGetBit(bitmapSrc,
xSrc, ySrc));
}
```

// _RotateBitmap90 and _RotateBitmap270 omitted for brevity

_GetEdgeInfo

// get info about the edges of the specified piece

```
static void _GetEdgeInfo(PUZ *ppuz, PCE *ppce)
{
  ROT *prot = &ppce->arot[0];
  BitMap *pbitmap = &prot->bitmapEdge;
  Rect bounds = pbitmap->bounds;
```

eat.

sleep.

Learn Cocoa®.

Big nerd ranch

Intensive Classes for Programmers
www.bignerdranch.com
404.210.5663

```
  ppce->cEdgeSide = 0;
  for (int idir = 0; idir < DIM(s_adirEdge); ++idir)
  {
    const DIR dir = s_adirEdge[idir];

    int x, y;
    int zFirst, zLim, *pz;

    if (dir.dx != 0)
    {
      pz = &x;
      zFirst = bounds.left;
      zLim = bounds.right;
      y = dir.dx > 0 ? bounds.top : bounds.bottom - 1;
    }
    else
    {
      pz = &y;
      zFirst = bounds.top;
      zLim = bounds.bottom;
      x = dir.dy > 0 ? bounds.left : bounds.right - 1;
    }

    // scan to see if this side looks like an edge piece
    EDGE *pedge = &prot->aedge[idir];
    pedge->fEdge = fFalse;
    int dzEdge = zLim - zFirst;
    if (dzEdge < 0)
      dzEdge = -dzEdge;

    int cpixEdge = 0;
    for (*pz = zFirst ; *pz != zLim; *pz += 1)
    {
      if (!_FGetBit(*pbitmap, x, y))
      {
        cpixEdge = 0;
        continue;
      }
      if (cpixEdge++ > 0)
      {
        if (cpixEdge == dzEdge/2)
          prot->aedge[idir].fEdge = fTrue;
      }
      else if (prot->aedge[idir].fEdge)
      {
        prot->aedge[idir].fEdge = fFalse;
        break;
      }
    }
    if (prot->aedge[idir].fEdge)
      ++ppce->cEdgeSide;
  }

  // copy the edge information to the other rotations
  for (int irot = 1; irot < DIM(ppce->arot); ++irot)
    for (int iedge = 0; iedge < 4; ++iedge)
      ppce->arot[irot].aedge[iedge] =
                prot->aedge[(iedge + irot) % 4];

}
```

_FindCorners

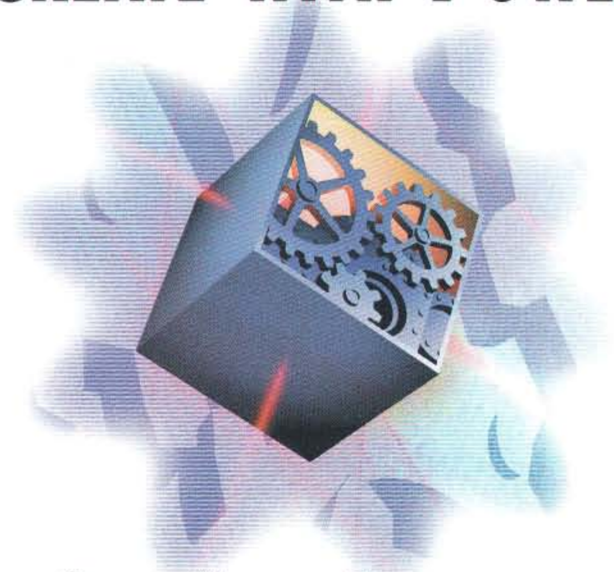
```
// find a reasonable guess for the corners of the piece stored in the bitmap
static void _FindCorners(const BitMap &bitmap,
            PT *pptTop, PT *pptBottom)
{
  Rect rectBounds = bitmap.bounds;

  pptTop->x = -1;
  for (int d = 0; pptTop->x == -1; ++d)
  {
    PT pt;
    for (pt.x = rectBounds.right-1, pt.y = rectBounds.top+d;
      pt.y < rectBounds.bottom && pt.x >= rectBounds.left;
      --pt.x, ++pt.y)
    {
      if (_FGetBit(bitmap, pt.x, pt.y))
      {
        pptTop->x = _XFirstBit(bitmap, pptTop->y = ++pt.y);
        break;
      }
    }
  }

  pptBottom->x = -1;
  for (int d = 0; pptBottom->x == -1;++d)
  {
    PT pt;
    for (pt.x = rectBounds.right-1, pt.y =
rectBounds.bottom-1-d;
      pt.y >= rectBounds.top && pt.x >= rectBounds.left;
      --pt.x, --pt.y)
    {
      if (_FGetBit(bitmap, pt.x, pt.y))
      {
        pptBottom->x = _XFirstBit(bitmap, pptBottom->y = --
pt.y);
        break;
      }
    }
  }
}
```

_CptGetEdgeShape

```
// return up to three "interesting" points along the right edge of the piece
// stored in bitmap
static int _CptGetEdgeShape(const BitMap &bitmap,
        PT paptCheck[3])
{
  // first, find the corners, more or less
  PT ptCornerTop, ptCornerBottom;
  _FindCorners(bitmap, &ptCornerTop, &ptCornerBottom);
  int dyQuarterEdge = (ptCornerBottom.y - ptCornerTop.y)/4;

  int cptCheck = 0;
  Rect rectBounds = bitmap.bounds;
  PT ptIn = {zNil, zNil}, ptOut = {zNil, zNil};

  int iptIn = -1, iptOut = -1;
  int xIn = rectBounds.right, xOut = rectBounds.left-1;
  int fOutPrev = fFalse, fInPrev = fFalse;
  int yMidMin = ptCornerTop.y+1;
  int yMidLim = ptCornerBottom.y;

  PT aptJump[255];
  int cptJump = 0;
  PT ptPrev;
  ptPrev.x = _XFirstBit(bitmap, ptPrev.y = yMidMin);
  aptJump[cptJump++] = ptPrev;
  for (PT ptCur = ptPrev; ++ptCur.y < yMidLim; )
  {
    ptCur.x = _XFirstBit(bitmap, ptCur.y);

    if (ptCur.x == ptPrev.x)
      continue;

    int dx = ptCur.x - ptPrev.x;
    int fIn = dx < 0;
    int fOut = !fIn;

    int yPeak;
    if (fOut && fInPrev)
    {
      // we have local min!
      if (ptPrev.x < xIn
        && _FMiddleish(ptPrev.y, ptCur.y, yMidMin, yMidLim,
&yPeak)
        )
      {
        ptIn.x = ptPrev.x;
        ptIn.y = yPeak;
        xIn = ptPrev.x;
        iptIn = cptJump-1;
      }
    }
    else if (fIn && fOutPrev)
    {
      // we have a local max!
      if (ptPrev.x > xOut
        && _FMiddleish(ptPrev.y, ptCur.y, yMidMin, yMidLim,
&yPeak)
        )
      {
        ptOut.x = ptPrev.x;
        ptOut.y = yPeak;
        xOut = ptPrev.x;
        iptOut = cptJump-1;
      }
    }

    aptJump[cptJump++] = ptCur;
    ptPrev = ptCur;
    fInPrev = fIn;
    fOutPrev = fOut;
  }

  if (ptIn.x != zNil)
  {
    int iptPrev, iptNext, dx;

    for (iptPrev = iptIn; ; )
    {
      if (--iptPrev < 0)
        goto LCheckOut;
      dx = aptJump[iptPrev+1].x - aptJump[iptPrev].x;
      if (dx < -1)
        break;
      if (dx > 0)
        if (dx > 1 || aptJump[iptPrev+1].x > ptIn.x+1)
          goto LCheckOut;
    }
    for (iptNext = iptIn; ; )
    {
      if (++iptNext >= cptJump)
        goto LCheckOut;
      dx = aptJump[iptNext].x - aptJump[iptNext-1].x;
      if (dx > 1)
        break;
      if (dx < 0)
        if (dx < -1 || aptJump[iptNext].x < ptIn.x-1)
          goto LCheckOut;
    }

    // there has to be some bump to qualify as an innie
    if (aptJump[iptNext].y - (aptJump[iptPrev+1].y-1) >
                      dyQuarterEdge*2)
      goto LSmooth;

    paptCheck[cptCheck].x = aptJump[iptPrev].x;
    paptCheck[cptCheck++].y = aptJump[iptPrev+1].y-1;
    paptCheck[cptCheck++] = ptIn;
    paptCheck[cptCheck++] = aptJump[iptNext];
    goto LRet;
  }

LCheckOut:
  if (ptOut.x != zNil && (ptIn.x == zNil ||
        ptOut.x == rectBounds.right-1))
  {
    int iptPrev, iptNext, dx;

    for (iptPrev = iptOut; ; )
    {
      if (--iptPrev < 0)
        goto LSmooth;
      dx = aptJump[iptPrev+1].x - aptJump[iptPrev].x;
      if (dx > 1)
        break;
      if (dx < 0)
        if (dx < -1 || aptJump[iptPrev+1].x < ptOut.x-1)
          goto LSmooth;
    }
    for (iptNext = iptOut; ; )
    {
      if (++iptNext >= cptJump)
        goto LSmooth;
      dx = aptJump[iptNext].x - aptJump[iptNext-1].x;
      if (dx < -1)
        break;
      if (dx > 0)
        if (dx > 1 || aptJump[iptNext].x > ptOut.x+1)
          goto LSmooth;
    }

    // there has to be some bump to qualify as an outie
    if (aptJump[iptNext].y - (aptJump[iptPrev+1].y-1) >
```

THE #1 RELATIONAL DATABASE ON MAC OS X

CREATE WITH POWER™



SHIP YOUR SOLUTIONS WITH OPENBASE SQL

HIGH-PERFORMANCE RELATIONAL DATABASE

OpenBase offers embedded licensing so you can ship your software products with confidence, knowing that they are powered by the #1 relational database server on Mac OS X.

Contact us at info@openbase.com for details!

What makes OpenBase SQL the best choice for your applications?

- Unique fault-tolerant data protection
- Proven scalable high-end performance
- Fast, easy database design and administration
- Full range of affordable solutions

Visit our Web site for more of the innovative features that make OpenBase SQL the database of choice for developers.

www.openbase.com

OPENBASE® INTERNATIONAL

```
            dyQuarterEdge*2)
        goto LSmooth;

      paptCheck[cptCheck].x = aptJump[iptPrev].x;
      paptCheck[cptCheck++].y = aptJump[iptPrev+1].y-1;
      paptCheck[cptCheck++] = ptOut;
      paptCheck[cptCheck++] = aptJump[iptNext];
      goto LRet;
    }

LSmooth:
  if (ptIn.x == zNil && ptOut.x == zNil)
    {
      int yWant = ptCornerTop.y + dyQuarterEdge;
      int ipt = 1;
      while (ipt < cptJump && aptJump[ipt].y <= yWant)
        ++ipt;
      paptCheck[cptCheck].y = yWant;
      paptCheck[cptCheck++].x = aptJump[ipt-1].x;

      yWant = ptCornerBottom.y - dyQuarterEdge;
      ipt = cptJump-1;
      while (aptJump[ipt].y > yWant)
        --ipt;

      paptCheck[cptCheck].y = yWant;
      paptCheck[cptCheck++].x = aptJump[ipt].x;
    }
  else
    {
      if (ptIn.x != zNil)
        paptCheck[cptCheck++] = ptIn;
      if (ptOut.x != zNil)
        paptCheck[cptCheck++] = ptOut;
    }

LRet:
  return cptCheck;
}
```

```
// determine if any element of the segment [x1, x2) is within the
// the range [xFirst, xLim). If not, return fFalse. If so,
// set *pxMid to a pixel inside the both ranges, as near the midpoint of
// [xFirst, xLim) as possible
static int _FMiddleish(int x1, int x2, int xFirst, int xLim,
int *pxMid)
{
  // calculate the midpoint and then the limits of the center segment
  int xMid = (xFirst + xLim)/2;

  if (x2 <= xFirst || xLim <= x1)
    return fFalse;

  if (x1 <= xMid && xMid < x2)
    *pxMid = xMid;
  else if (x1 > xMid)
    *pxMid = x1;
  else
    *pxMid = x2-1;
  return fTrue;
}

// find the rightmost set bit in the specified row of the bitmap
static int _XFirstBit(const BitMap &bitmap, int y)
{
  int x;
  for (x = bitmap.bounds.right-1; x >= bitmap.bounds.left; --
x)
    if (_FGetBit(bitmap, x, y))
      break;
  return x;
}

// MusecFromTime omitted for brevity
```

BMS

THE LAW OFFICE OF BRADLEY M. SNIDERMAN

Need help safeguarding your software?

If you're developing software, you need your valuable work protected with trademark and copyright registration, as well as Non Disclosure Agreements.

Then, when you are ready to sell it, you can protect yourself further with a licensing agreement.

I am an attorney practicing in Intellectual Property, Business Formations, Corporate, Commercial and Contract law.

Please give me a call or an e-mail. Reasonable fees.

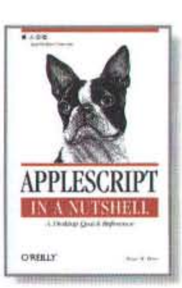
23679 Calabasas Rd. #558 • Calabasas, CA 91302
PHONE 818-222-0365 FAX 818-591-1038 EMAIL brad@sniderman.com

Trying to keep up with OS X?

WE CAN GET YOU ON THE INSIDE TRACK.

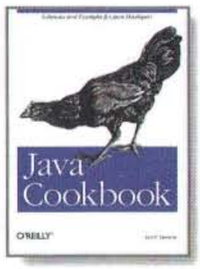
WE'RE TRUE INSIDERS.

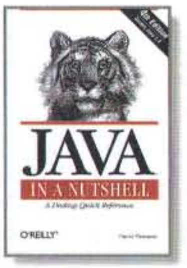
O'Reilly and Apple have joined forces to bring you unparalleled access to the experts developing OS X. Get on the inside track (and stay there) with our OS X books and the O'Reilly Network's Mac DevCenter at www.oreillynet.com/mac.

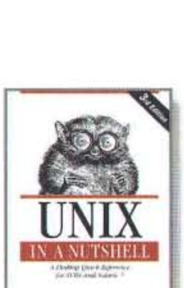


WE KEEP PACE WITH JAVA.

Our Java books—and our online site for Java developers, on.java.com—offer the most comprehensive and up-to-date information available to help you keep pace, too.







WE KNOW UNIX— INSIDE AND OUT.

We cut our teeth on Unix: O'Reilly has been publishing the most respected and useful books on Unix for well over a decade, and we're still at the head of the pack.

O'REILLY®

WWW.OREILLY.COM • 800-998-9938

**ADVERTISER/
PRODUCT LIST**

List of Advertisers

List of Products

The index on this page is provided as a service to our readers. The publisher does not assume any liability for errors or omissions.

For developers under pressure to manage source code and do more in less time, Perforce's Fast Software Configuration Management System is the must-have tool.

With rival SCM systems, the only way to quicken the pace is to cut corners - but in the long run you pay the price with missed deadlines, uncertain contents, buggy releases and no way back to previous builds.

With Perforce, the fast way is always the right way. Install it fast, learn it fast, execute operations fast. With other SCM systems, developers face an unpleasant choice: do it the right way or do it the fast way. Perforce's speed and reliability mean fast is right. See how Perforce compares with other leading SCM systems at http://www.perforce.com/perforce/reviews.html

Run at full speed even with hundreds of users and millions of files. At the core of Perforce lies a relational database with well-keyed tables, so simple operations can be accomplished in near-zero time. Larger operations (like labeling a release and branching) are translated into keyed data access, giving Perforce the scalability that big projects require.

Work anywhere. Perforce is efficient over high-latency networks such as WANs, the Internet and even low-speed dial-up connections. Requiring only TCP/IP, Perforce makes use of a well-tuned streaming message protocol for synchronizing client workspace with server repository contents.

Develop and maintain multiple codelines. Perforce Inter-File Branching™ lets you merge new features and apply fixes between codelines. Smart metadata keeps track of your evolving projects even while they develop in parallel.

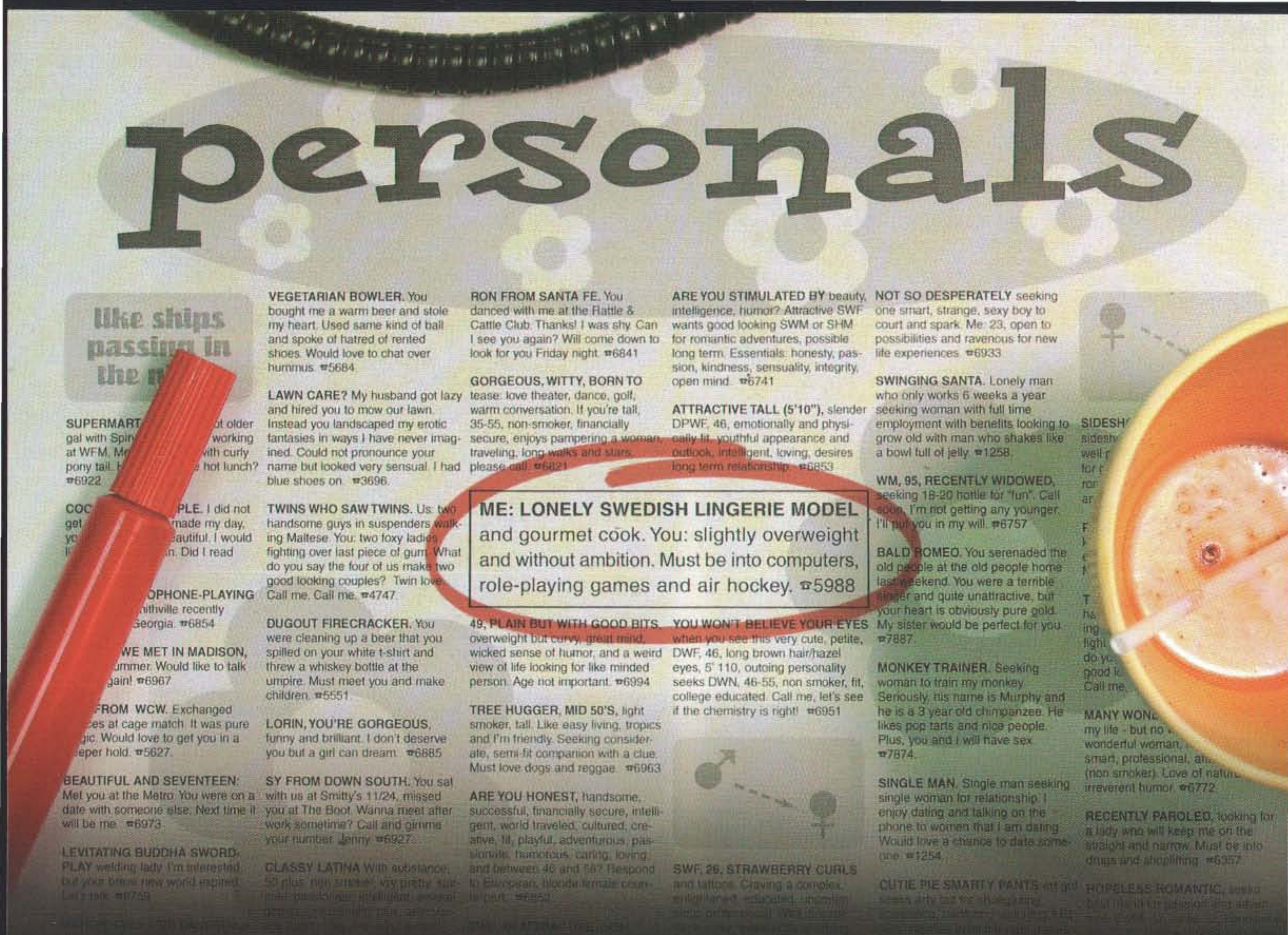
Truly cross platform. Perforce runs on more than 50 operating systems, including Windows and nearly every UNIX variation, from Linux and Mac OS X to AS400 and more.

Integrate with leading IDEs and defect trackers: Visual Studio.NET®, Visual C++®, Visual Basic®, JBuilder®, CodeWarrior®, TeamTrack®, Bugzilla™, ControlCenter®, DevTrack® packages, and more.

PERFORCE
SOFTWARE

Fast Software Configuration Management www.perforce.com

Download your free 2-user, non-expiring, full-featured copy now from www.perforce.com
Free (and friendly) technical support is on hand to answer any and all evaluation questions.



personals

like ships passing in the n

SUPERMART t older gal with Spin working at WFM. Me ith curly pony tail. hot lunch? ☎6922

COC PLE. I did not get made my day, yo autiful. I would l n. Did I read

OPHONE-PLAYING ithville recently Georgia. ☎6854

WE MET IN MADISON, ummer. Would like to talk gain! ☎6967

FROM WCW. Exchanged es at cage match. It was pure gic. Would love to get you in a eper hold. ☎5627.

BEAUTIFUL AND SEVENTEEN: Met you at the Metro. You were on a date with someone else. Next time it will be me. ☎6973.

LEVITATING BUDDHA SWORD-PLAY welding lady I'm interested but your brave new world inspired. Let's talk. ☎[illegible]

VEGETARIAN BOWLER. You bought me a warm beer and stole my heart. Used same kind of ball and spoke of hatred of rented shoes. Would love to chat over hummus. ☎5684

LAWN CARE? My husband got lazy and hired you to mow our lawn. Instead you landscaped my erotic fantasies in ways I have never imagined. Could not pronounce your name but looked very sensual. I had blue shoes on. ☎3696.

TWINS WHO SAW TWINS. Us: two handsome guys in suspenders walking Maltese. You: two foxy ladies fighting over last piece of gum. What do you say the four of us make two good looking couples? Twin love. Call me. Call me. ☎4747

DUGOUT FIRECRACKER. You were cleaning up a beer that you spilled on your white t-shirt and threw a whiskey bottle at the umpire. Must meet you and make children. ☎5551

LORIN, YOU'RE GORGEOUS, funny and brilliant. I don't deserve you but a girl can dream. ☎6885

SY FROM DOWN SOUTH. You sat with us at Smitty's 11/24, missed you at The Boot. Wanna meet after work sometime? Call and gimme your number. Jenny ☎6927

CLASSY LATINA With substance, [illegible]

RON FROM SANTA FE. You danced with me at the Rattle & Cattle Club. Thanks! I was shy. Can I see you again? Will come down to look for you Friday night. ☎6841

GORGEOUS, WITTY, BORN TO tease: love theater, dance, golf, warm conversation. If you're tall, 35-55, non-smoker, financially secure, enjoys pampering a woman, traveling, long walks and stars, please call ☎6821

ARE YOU STIMULATED BY beauty, intelligence, humor? Attractive SWF wants good looking SWM or SHM for romantic adventures, possible long term. Essentials: honesty, passion, kindness, sensuality, integrity, open mind. ☎6741

ATTRACTIVE TALL (5'10"), slender DPWF, 46, emotionally and physically fit, youthful appearance and outlook, intelligent, loving, desires long term relationship. ☎6853

ME: LONELY SWEDISH LINGERIE MODEL and gourmet cook. You: slightly overweight and without ambition. Must be into computers, role-playing games and air hockey. ☎5988

49, PLAIN BUT WITH GOOD BITS. overweight but curvy, great mind, wicked sense of humor, and a weird view of life looking for like minded person. Age not important. ☎6994

TREE HUGGER, MID 50'S, light smoker, tall. Like easy living, tropics and I'm friendly. Seeking considerate, semi-fit companion with a clue. Must love dogs and reggae. ☎6963

ARE YOU HONEST, handsome, successful, financially secure, intelligent, world traveled, cultured, creative, fit, playful, adventurous, passionate, humorous, caring, loving and between 46 and 56? Respond to European, blonde female counterpart. ☎[illegible]

YOU WON'T BELIEVE YOUR EYES when you see this very cute, petite, DWF, 46, long brown hair/hazel eyes, 5' 110, outoing personality seeks DWN, 46-55, non smoker, fit, college educated. Call me, let's see if the chemistry is right! ☎6951

SWF, 26, STRAWBERRY CURLS and tattoos. Craving a complex, enlightened, educated, [illegible]

NOT SO DESPERATELY seeking one smart, strange, sexy boy to court and spark. Me: 23, open to possibilities and ravenous for new life experiences. ☎6933

SWINGING SANTA. Lonely man who only works 6 weeks a year seeking woman with full time employment with benefits looking to grow old with man who shakes like a bowl full of jelly. ☎1258.

WM, 95, RECENTLY WIDOWED, seeking 18-20 hottie for "fun". Call soon, I'm not getting any younger. I'll put you in my will. ☎6757

BALD ROMEO. You serenaded the old people at the old people home last weekend. You were a terrible singer and quite unattractive, but your heart is obviously pure gold. My sister would be perfect for you. ☎7887.

MONKEY TRAINER. Seeking woman to train my monkey. Seriously, his name is Murphy and he is a 3 year old chimpanzee. He likes pop tarts and nice people. Plus, you and I will have sex ☎7874.

SINGLE MAN. Single man seeking single woman for relationship. I enjoy dating and talking on the phone to women that I am dating. Would love a chance to date someone. ☎1254.

CUTIE PIE SMARTY PANTS [illegible]

SIDESH sidesh well for ror ar

MANY WON my life - but no wonderful woman, smart, professional, a (non smoker). Love of nature irreverent humor. ☎6772

RECENTLY PAROLED, looking for a lady who will keep me on the straight and narrow. Must be into drugs and shoplifting. ☎6357

HOPELESS ROMANTIC, [illegible]

We're Easier.

Create anything from prototypes to full professional applications. Just drag and drop interface elements while REALbasic handles the details. You concentrate on what makes your stuff great — your ideas! REALbasic creates native compiled applications for Macintosh, Mac OS X and Windows without platform-specific adjustments. It's the powerful, easy-to-use tool for creating your own software. Each version of your software looks and works just as it should in each environment.

Complex problems shouldn't require complex solutions. The answer is REALbasic.

REALbasic 4.5 NEW VERSION

Come see us at Macworld in MacTech Central! Download a free demo. www.realbasic.com